Как прикрутить SQL к чему угодно при помощи Apache Calcite

Роман Кондаков, Querify Labs





Обо мне



Обо мне

- Работаю в Querify Labs
    - Консультируем по созданию СУБД и использованию Apache Calcite



Обо мне

- Работаю в Querify Labs
  - Консультируем по созданию СУБД и использованию Apache Calcite
- Ex-Yandex, недолго занимался Yandex Query Language

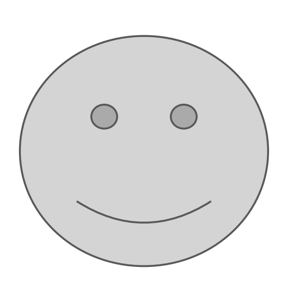


Обо мне

- Работаю в Querify Labs
  - Консультируем по созданию СУБД и использованию Apache Calcite
- Ex-Yandex, недолго занимался Yandex Query Language
- Ex-Gridgain, делал SQL-движок для Apache Ignite



SQL в Apache Ignite. Попытка №1







SQL в Apache Ignite. Попытка №1

put(Key, Value)

apache Ignite™



SQL в Apache Ignite. Попытка №1

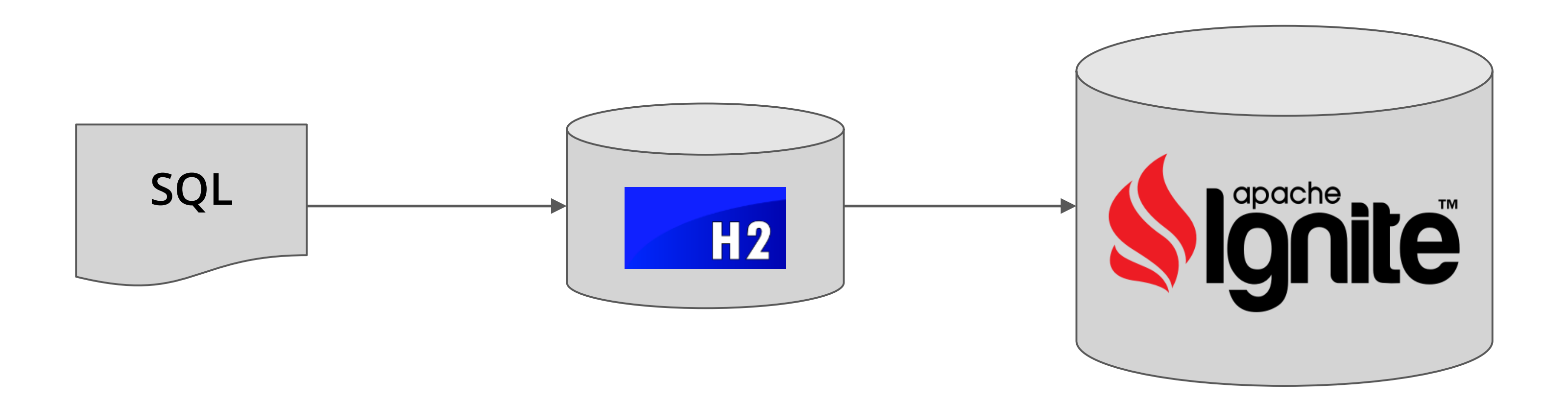
put(Key, Value)

get(Key)

apache Ignite™



SQL в Apache Ignite. Попытка №1

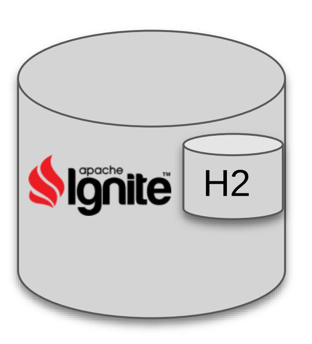




SQL в Apache Ignite. Попытка №1



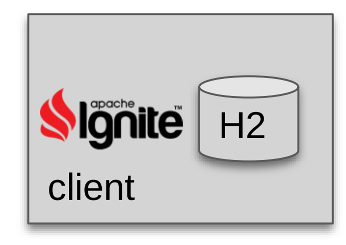






SQL в Apache Ignite. Попытка №1

```
SELECT AVG(salary) FROM employees
```









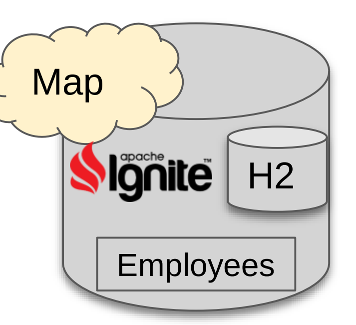
SQL в Apache Ignite. Попытка №1

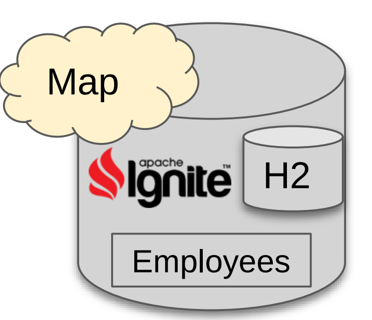
```
SELECT AVG(salary) FROM employees
```

=

```
Map query:
SELECT SUM(salary) as sum0,
COUNT(salary) as count0
FROM employees
```







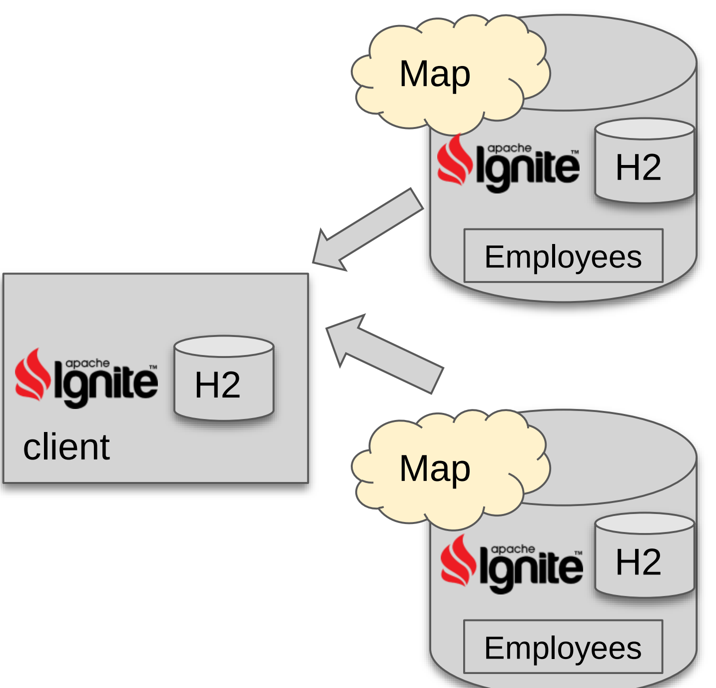


SQL в Apache Ignite. Попытка №1

```
SELECT AVG(salary) FROM employees
```

=

```
Map query:
SELECT SUM(salary) as sum0,
COUNT(salary) as count0
FROM employees
```





SQL в Apache Ignite. Попытка №1

```
SELECT AVG(salary) FROM employees
```

=

```
Map query:
SELECT SUM(salary) as sum0,
COUNT(salary) as count0
FROM employees
```

+

```
Reduce query:
SELECT SUM(sum0) / SUM(count0)
FROM resultTable
```

Map

apache Ignite H2

Employees

Reduce

apache Ignite H2

client

Map

apache Ignite H2

Employees



SQL в Apache Ignite. Попытка №1

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```



SQL в Apache Ignite. Попытка №1

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

=

```
X = SELECT AVG(emps.salary) FROM emps
```

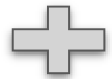


SQL в Apache Ignite. Попытка №1

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```



```
X = SELECT AVG(emps.salary) FROM emps
```



```
SELECT * FROM emps WHERE emps.salary = X
```



SQL в Apache Ignite. Попытка №1

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

=

```
X = SELECT AVG(emps.salary) FROM emps
```

= Map qry + Reduce qry

+

```
SELECT * FROM emps WHERE emps.salary = X
```

= Map qry + Reduce qry



SQL в Apache Ignite. Попытка №1

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

2 x Map + 2 x Reduce == Not OK

=

```
X = SELECT AVG(emps.salary) FROM emps
```

= Map qry + Reduce qry

+

```
SELECT * FROM emps WHERE emps.salary = X
```

= Map qry + Reduce qry



SQL в Apache Ignite. Попытка №1

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

=

```
X = SELECT AVG(emps.salary) FROM emps
```

= Map qry + Reduce qry

+

```
SELECT * FROM emps WHERE emps.salary = X
```

= Map qry + Reduce qry

2 x Map + 2 x Reduce == Not OK

1 x Map + 1 x Reduce == OK



Как починить SQL в Apache Ignite?



Как починить SQL в Apache Ignite?

Варианты:

- Усовершенствовать старый движок (мрачные перспективы)



Как починить SQL в Apache Ignite?

Варианты:

- Усовершенствовать старый движок (мрачные перспективы)
- Написать с нуля (долго и рискованно)



Как починить SQL в Apache Ignite?

Варианты:

- Усовершенствовать старый движок (мрачные перспективы)
- Написать с нуля (долго и рискованно)
- Поискать готовые решения (а вот это интересно)



Какое-такое готовое решение?



Какое-такое готовое решение?

Apache Calcite – фреймворк для создания SQL БД



Какое-такое готовое решение?

Apache Calcite – фреймворк для создания SQL БД

NoSQL + APACHE calcite™ = SQL



Какое-такое готовое решение?

Apache Calcite – фреймворк для создания SQL БД

NoSQL + APACHE calcite™ = SQL

- Как мы при помощи Calcite чиним SQL в Apache Ignite
- Внутренности Apache Calcite
- Нюансы для распределенных систем

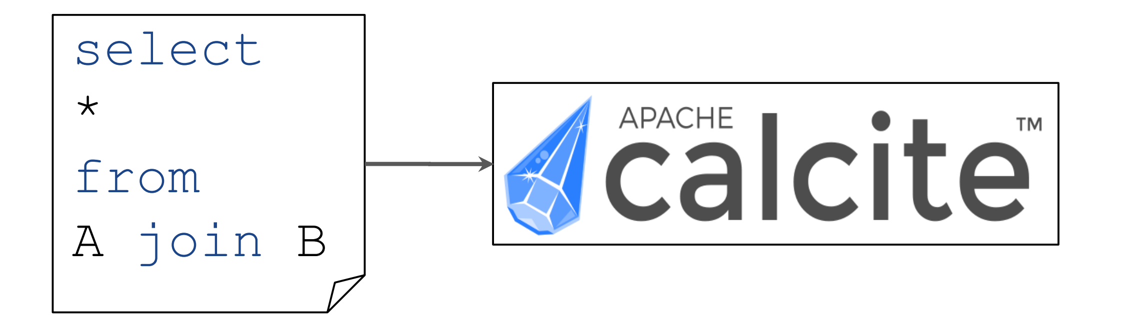


А как использовать Кальцит?

```
select
*
from
A join B
```



А как использовать Кальцит?





А как использовать Кальцит?





А как использовать Кальцит?

```
select
*
from
A join B
```

APACHE calcite™

Query plan

Join

Scan Scan

apache Ignite™



А как использовать Кальцит?

```
select
*
from
A join B
```

APACHE calcite ™

Query plan

Join

Scan Scan

Что угодно



Кто еще использует?

- Стартовал в начале 2000-х
- Apache с 2013 г.

Used by









Connects to





















А что внутри Кальцита?

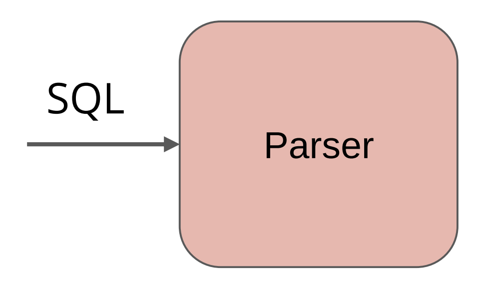


А что внутри Кальцита?

SQL

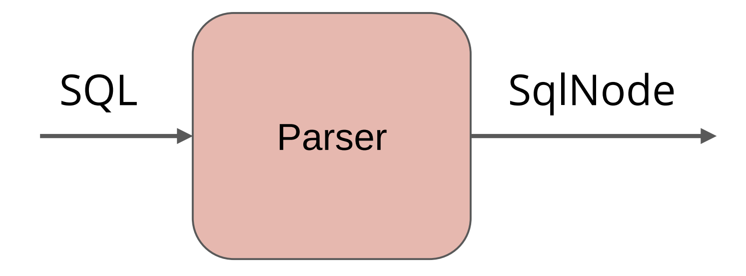


А что внутри Кальцита?



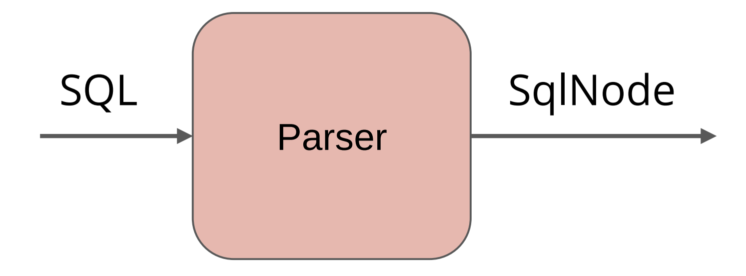


А что внутри Кальцита?



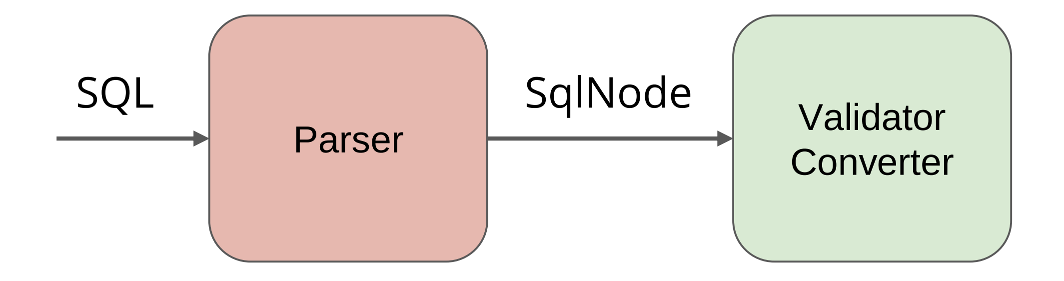


А что внутри Кальцита?



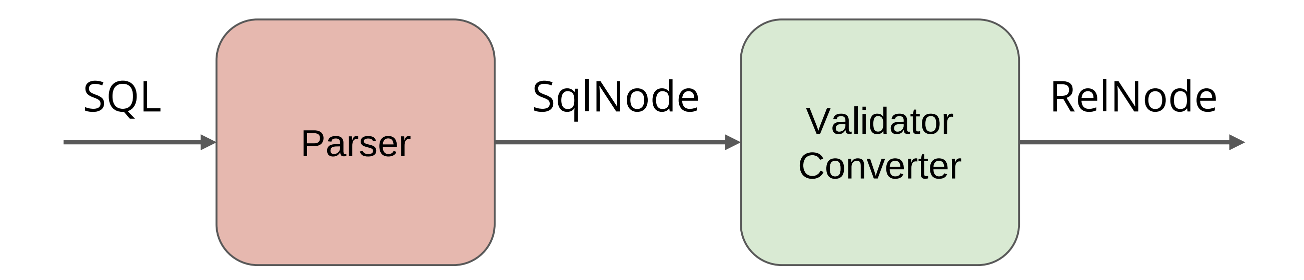


А что внутри Кальцита?



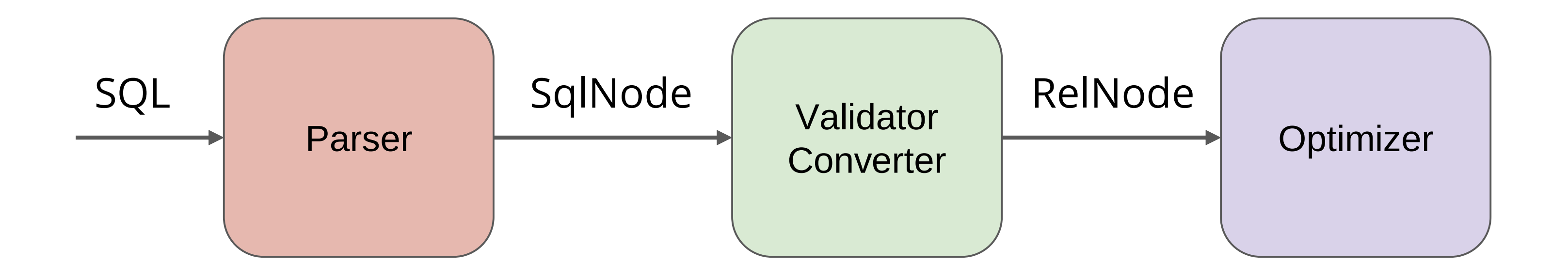


А что внутри Кальцита?



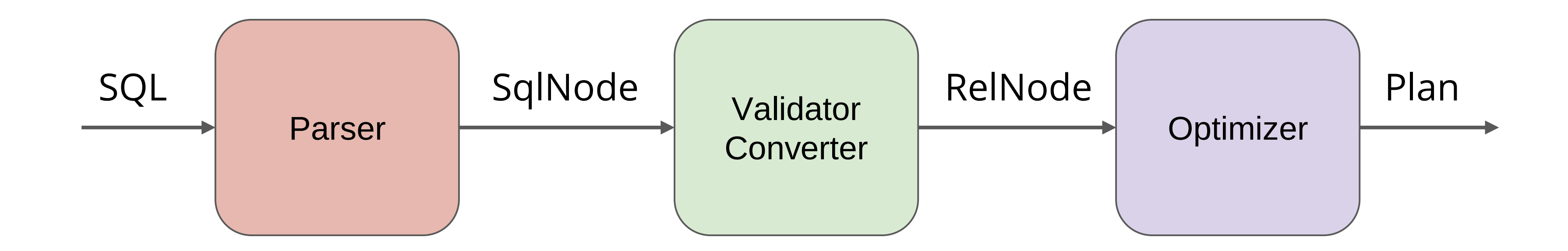


А что внутри Кальцита?





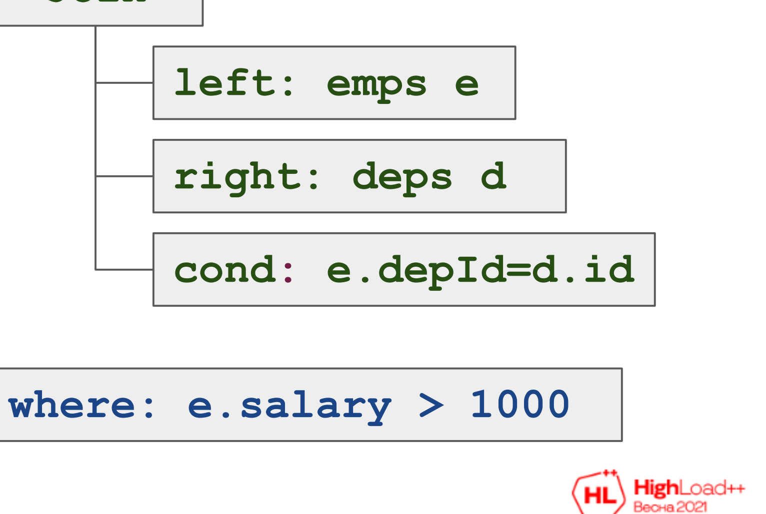
А что внутри Кальцита?





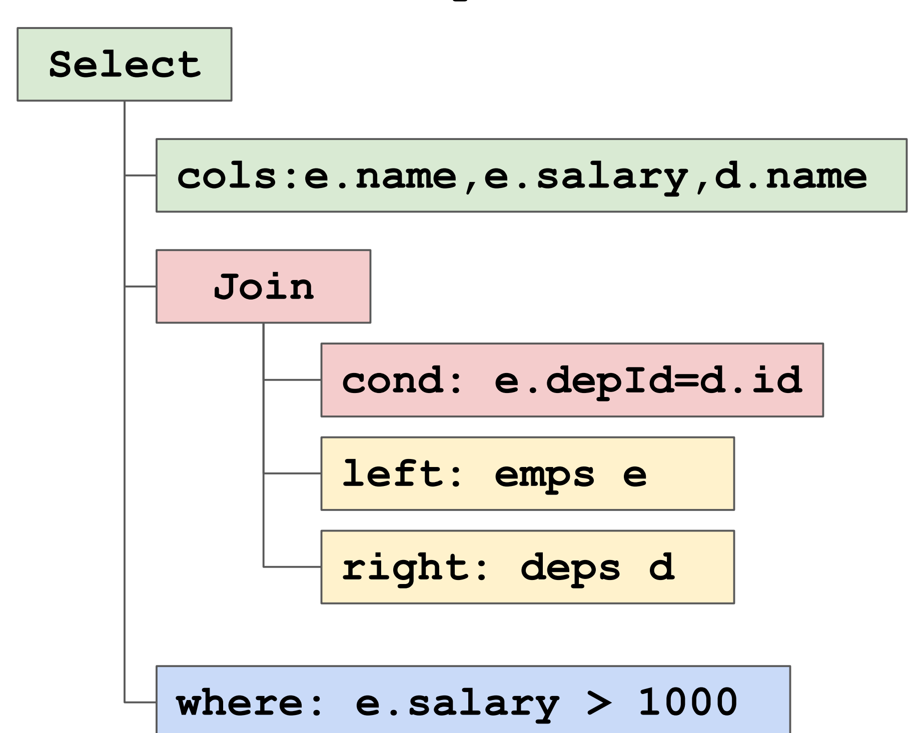
Парсер

```
SELECT
  e.name, e.salary, d.name
FROM
  emps e JOIN deps e
ON
  e.depId = d.id
WHERE
  e.salary > 1000
```



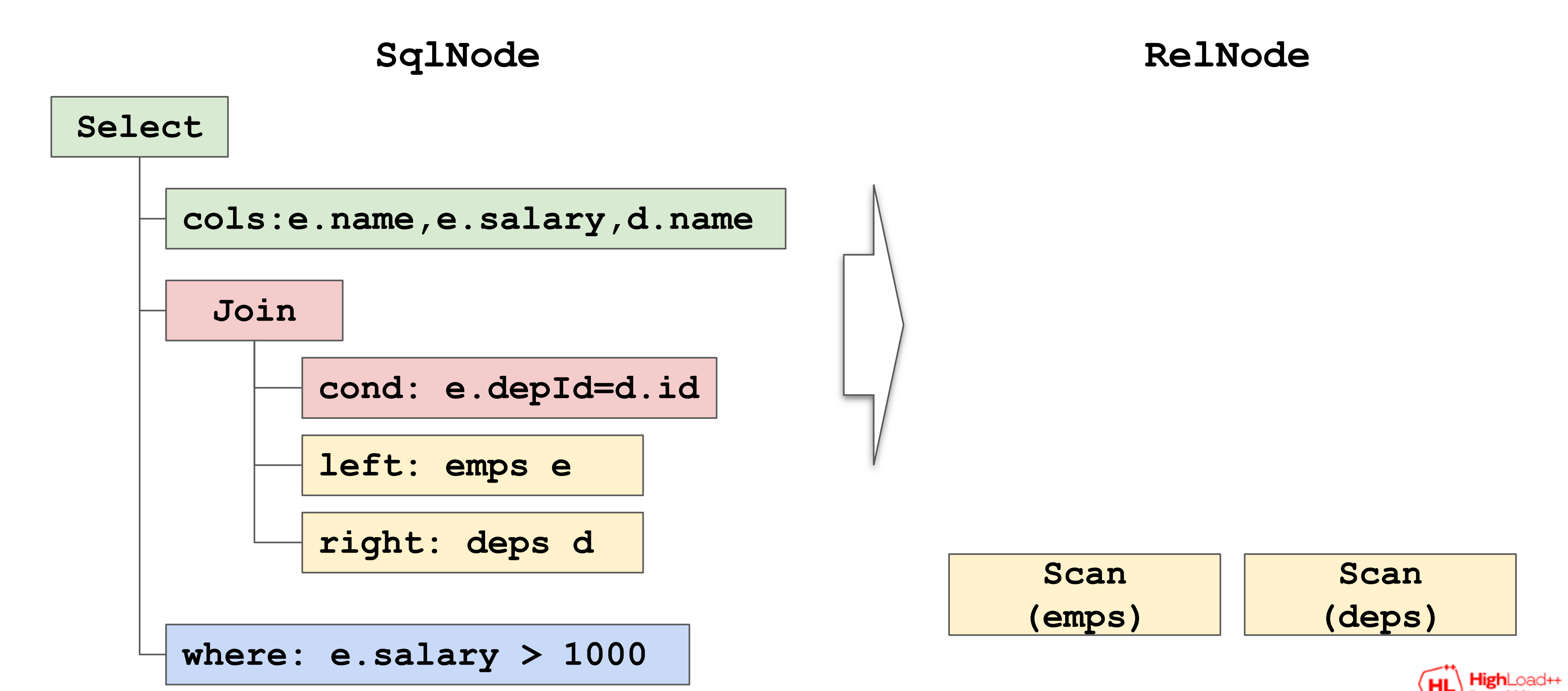


Конвертер

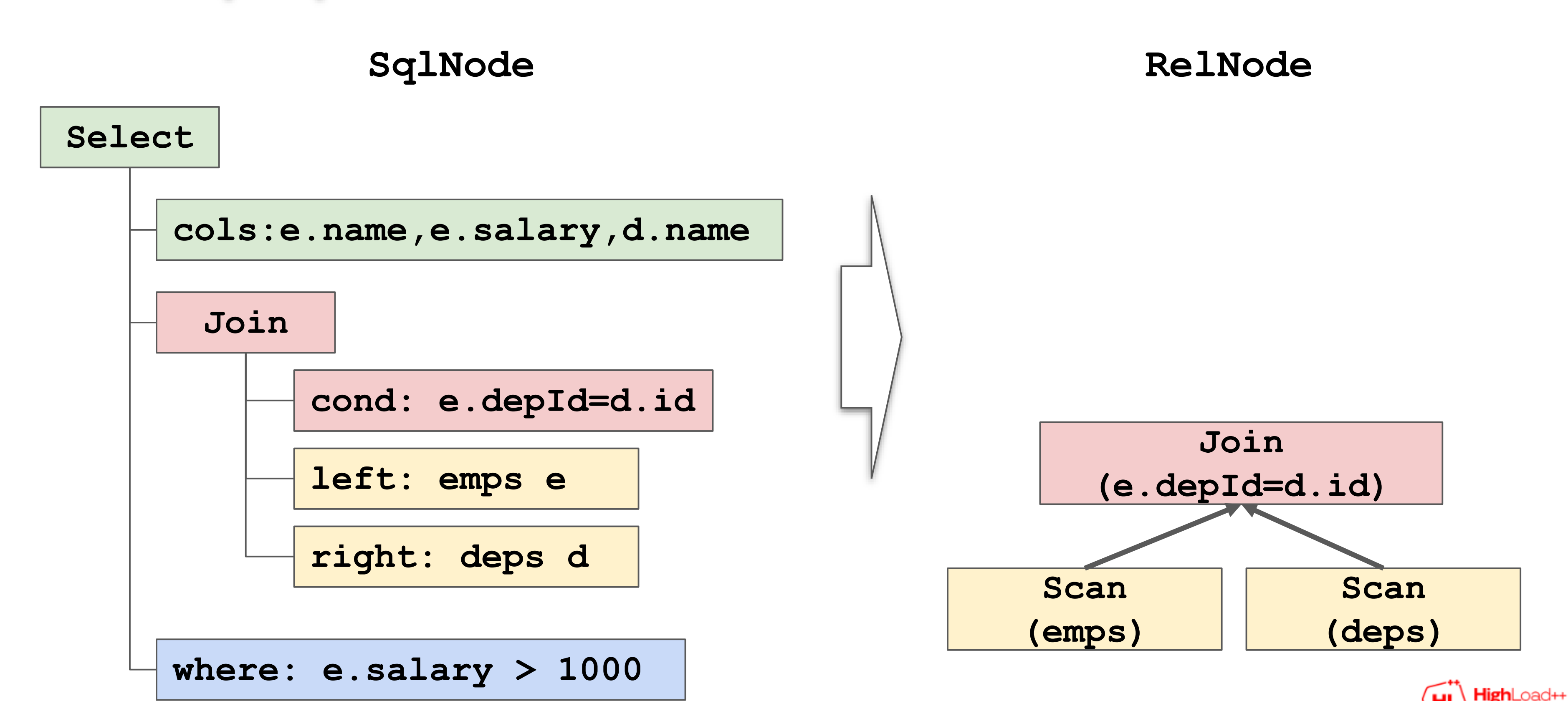

SqlNode
Select
cols:e.name,e.salary,d.name
Join
cond: e.depId=d.id
left: emps e
right: deps d
where: e.salary > 1000


HL
HighLoad++
Весна 2021

Конвертер


SqlNode
RelNode
Select
cols:e.name,e.salary,d.name
Join
cond: e.depId=d.id
left: emps e
right: deps d
where: e.salary > 1000
Scan
(emps)
Scan
(deps)
HL HighLoad++
Весна 2021

Конвертер


SqlNode
RelNode
Select
cols:e.name,e.salary,d.name
Join
cond: e.depId=d.id
left: emps e
right: deps d
where: e.salary > 1000
Join
(e.depId=d.id)
Scan
(emps)
Scan
(deps)
HL HighLoad++
Весна 2021

Конвертер

SqlNode

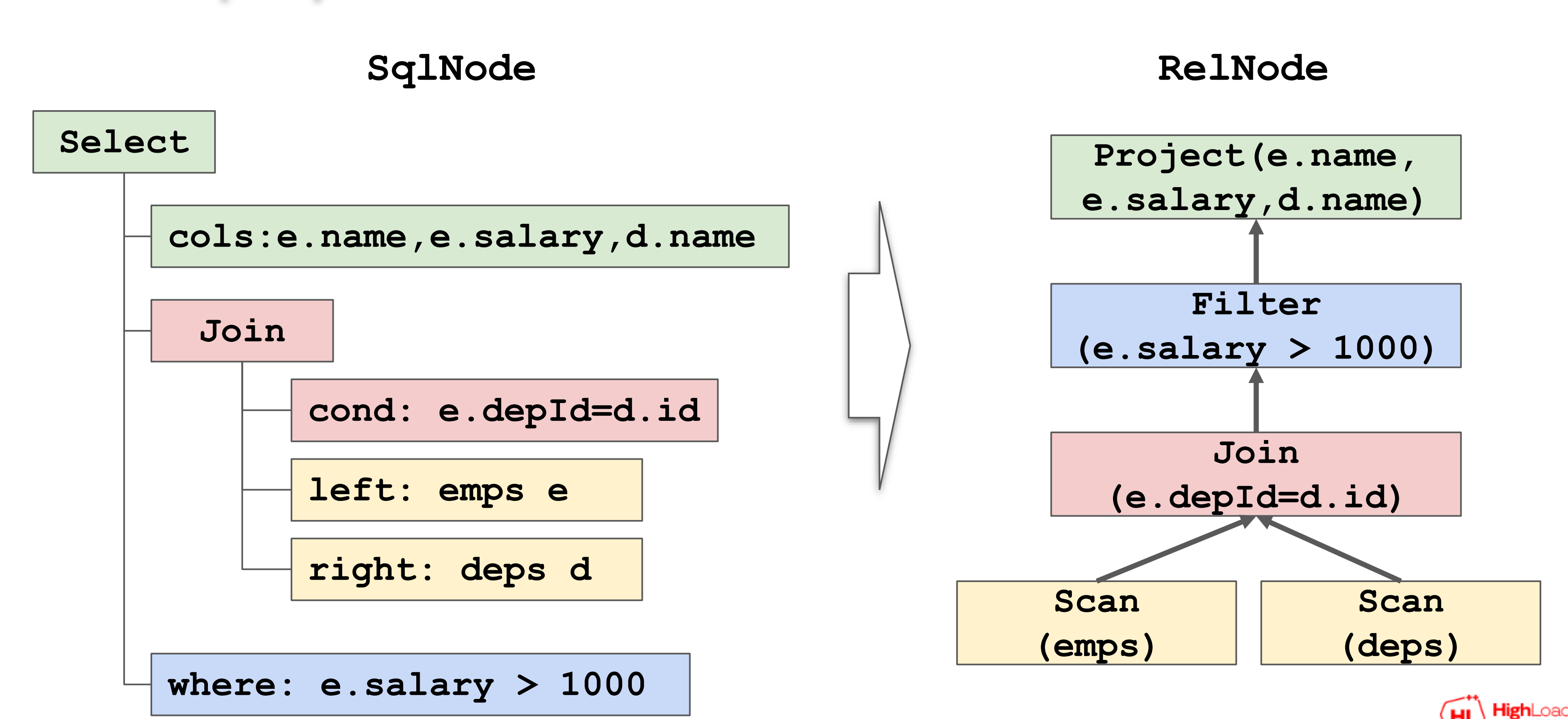
- Select
  - cols:e.name,e.salary,d.name
  - Join
    - cond: e.depId=d.id
    - left: emps e
    - right: deps d
  - where: e.salary > 1000

RelNode

- Filter (e.salary > 1000)
  - Join (e.depId=d.id)
    - Scan (emps)
    - Scan (deps)



Конвертер



SqlNode
RelNode
Select
cols:e.name,e.salary,d.name
Join
cond: e.depId=d.id
left: emps e
right: deps d
where: e.salary > 1000
Project(e.name,
e.salary,d.name)
Filter
(e.salary > 1000)
Join
(e.depId=d.id)
Scan
(emps)
Scan
(deps)
HL HighLoad++
Весна 2021

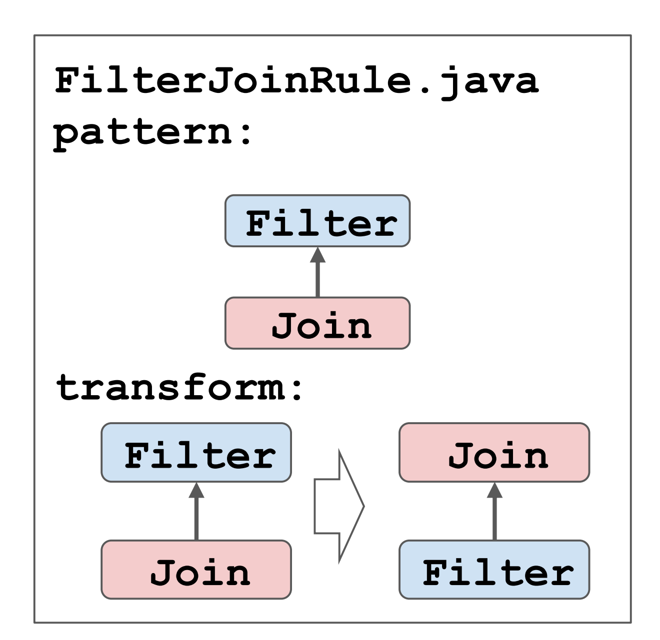
Оптимизатор и его правила

- Конфигурируется при помощи правил



Оптимизатор и его правила

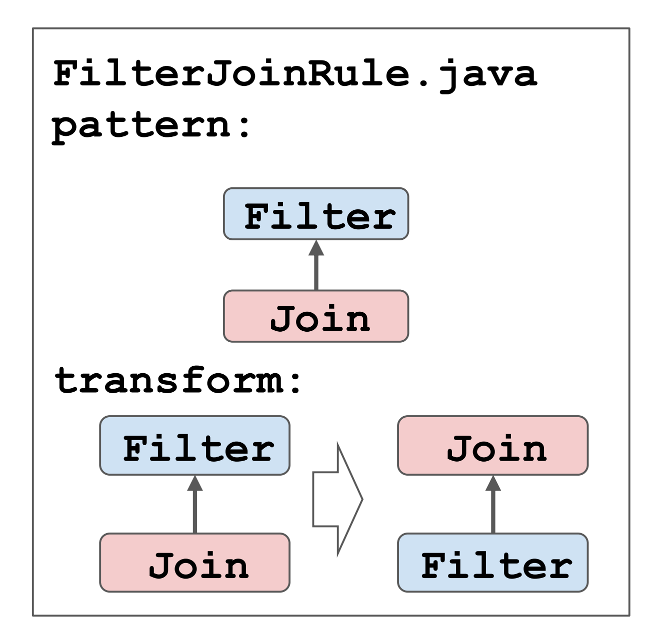
- Конфигурируется при помощи правил

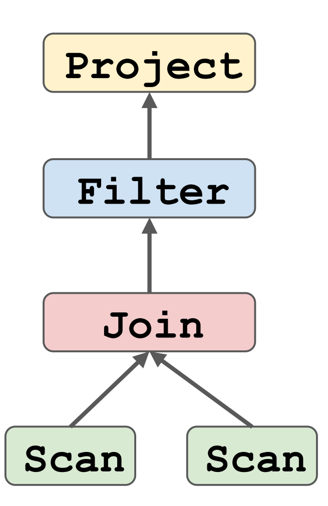




Оптимизатор и его правила

- Конфигурируется при помощи правил

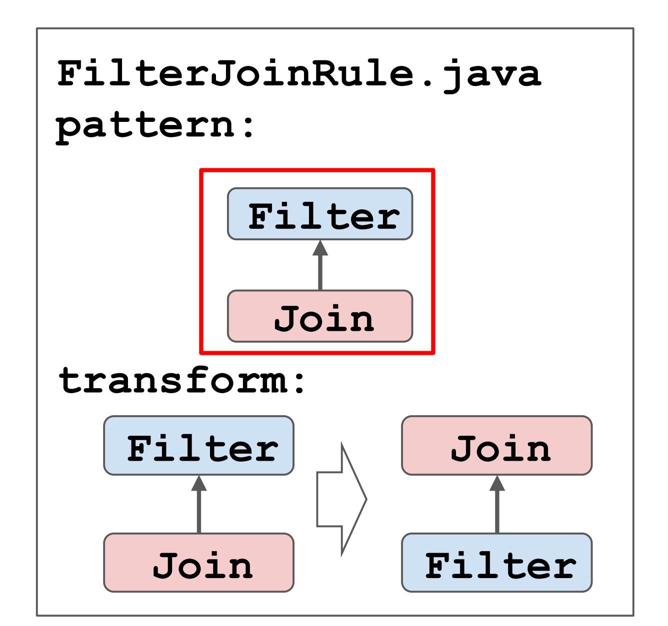


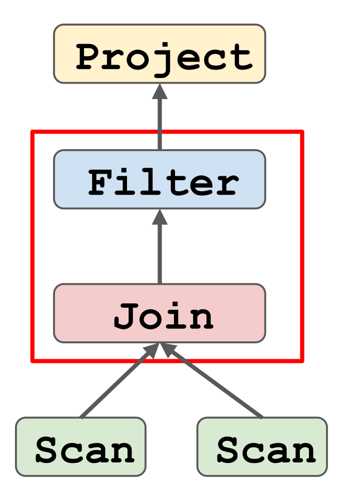




Оптимизатор и его правила

- Конфигурируется при помощи правил

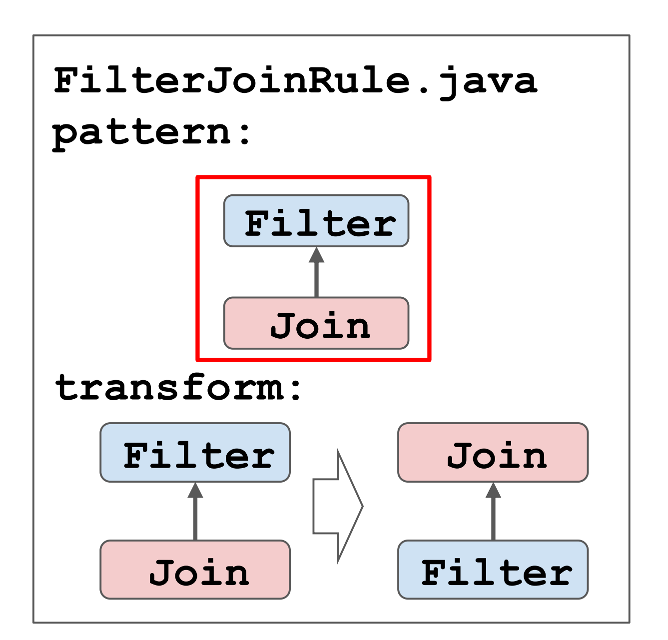


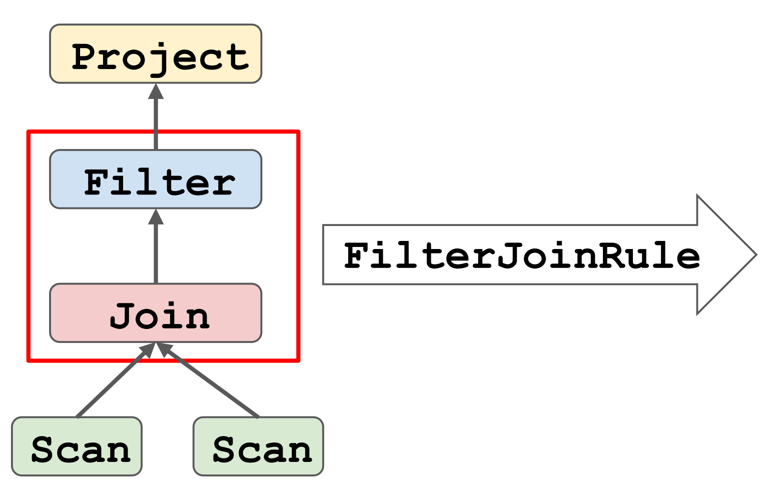




Оптимизатор и его правила

- Конфигурируется при помощи правил

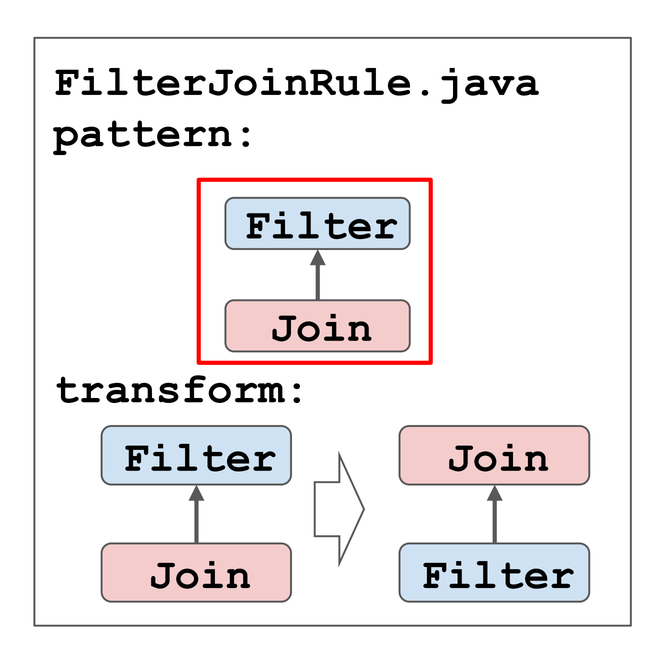


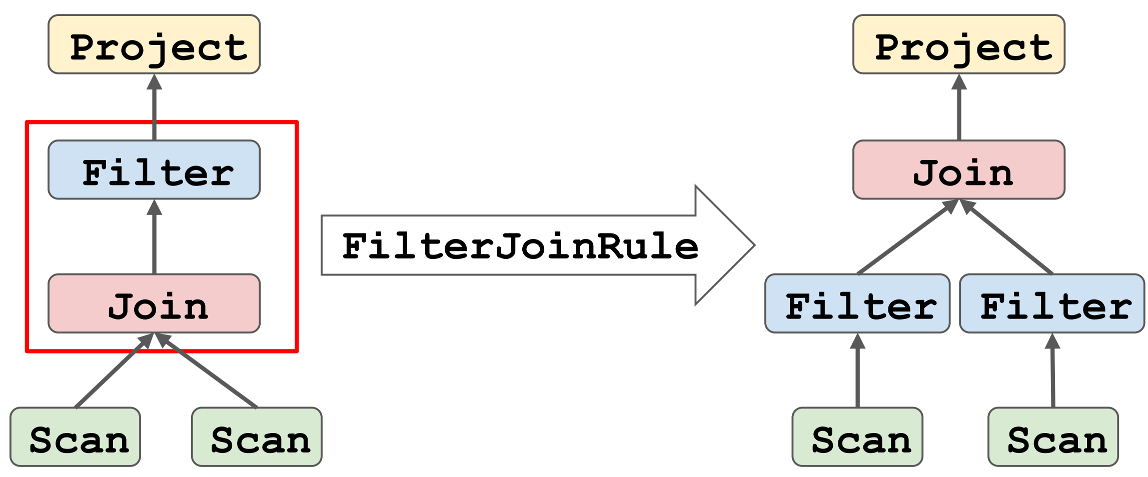




Оптимизатор и его правила

- Конфигурируется при помощи правил







HepPlanner – эвристический (rule-based) оптимайзер

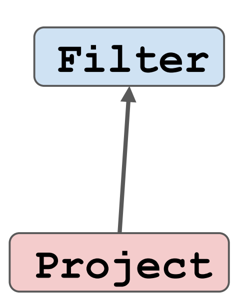
- Применяет правила, пока может (возможно зацикливание)
- Быстрый
- Используется для всегда оптимальных преобразований

```
while(canApplyRule()) {
        applyRule();
}
```



VolcanoPlanner – cost based optimizer

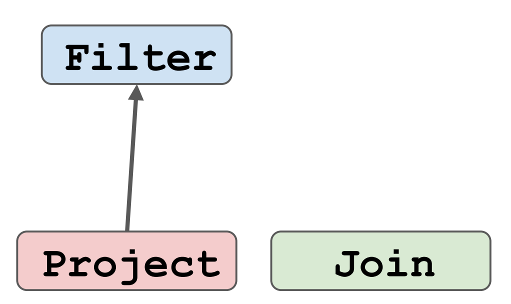
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

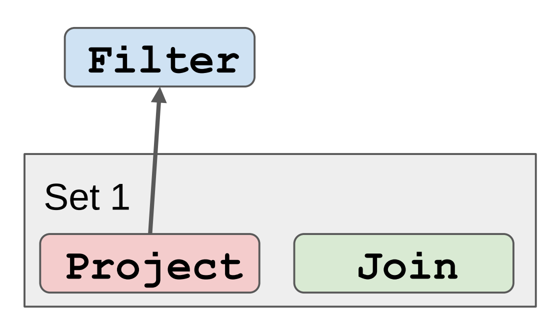
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

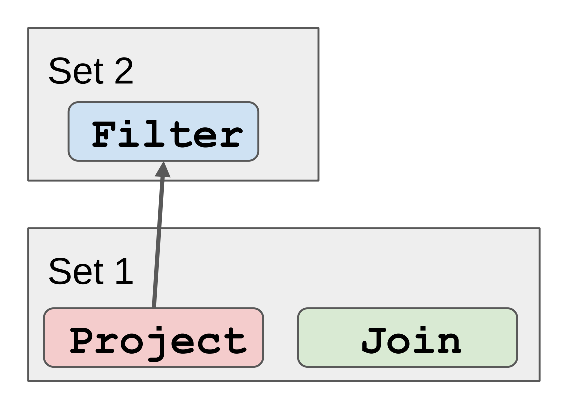
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

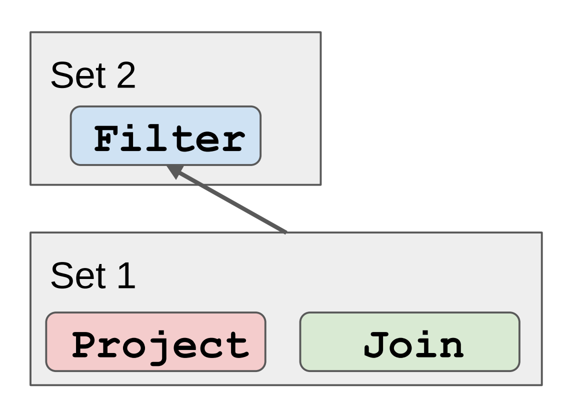
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

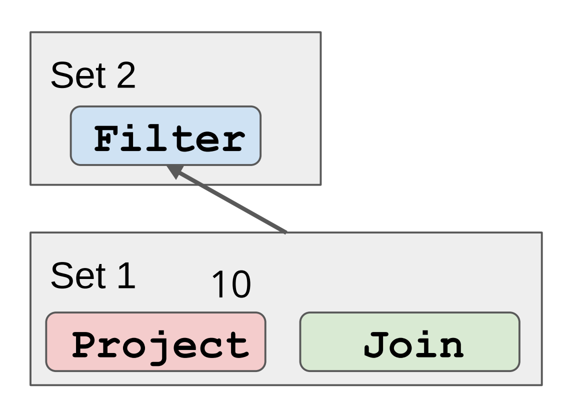
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

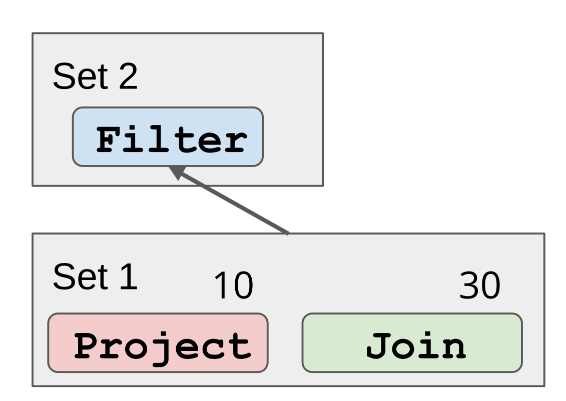
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

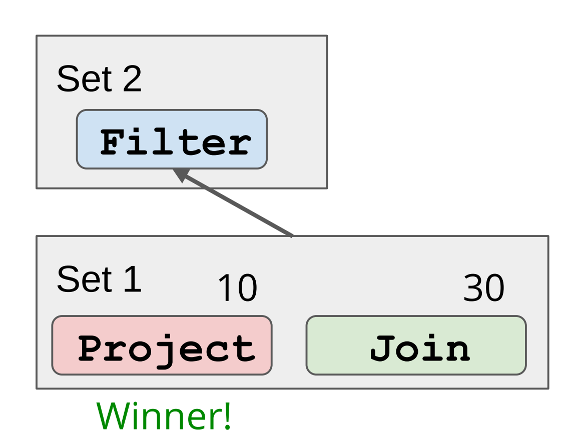
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

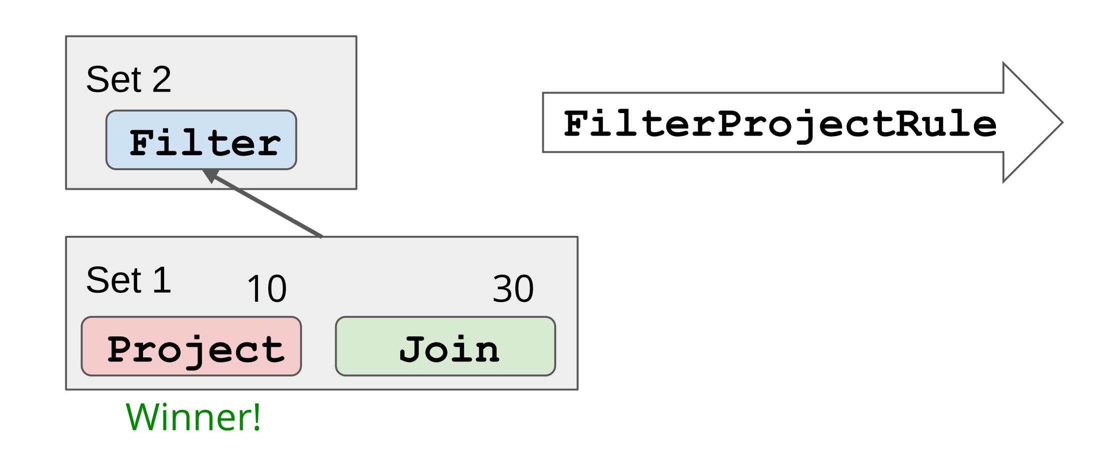
- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план





VolcanoPlanner – cost based optimizer

- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план



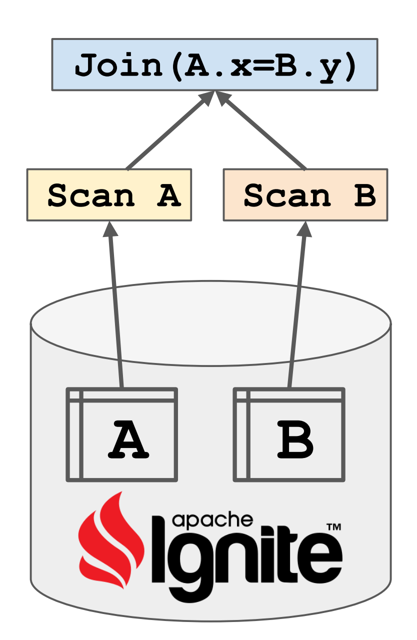
VolcanoPlanner – cost based optimizer

- Учитывает кост операторов
- Хранит все модификации дерева запроса в структуре MEMO
- Сначала применяет правила, потом ищет в MEMO лучший план



VolcanoPlanner – физическое планирование

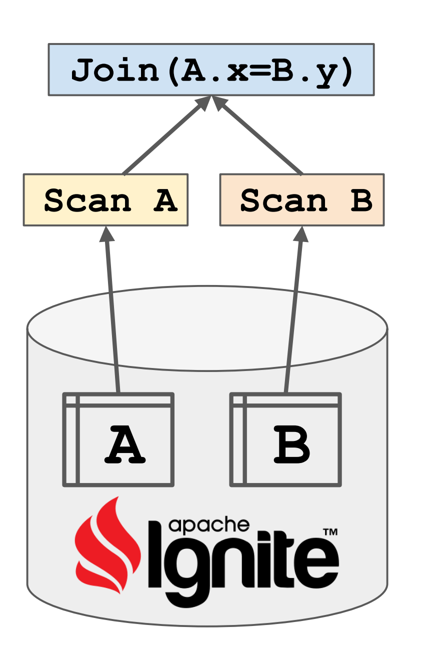
Logical plan





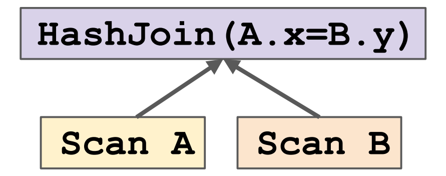
VolcanoPlanner – физическое планирование

Logical plan



HashJoinRule

Physical plan

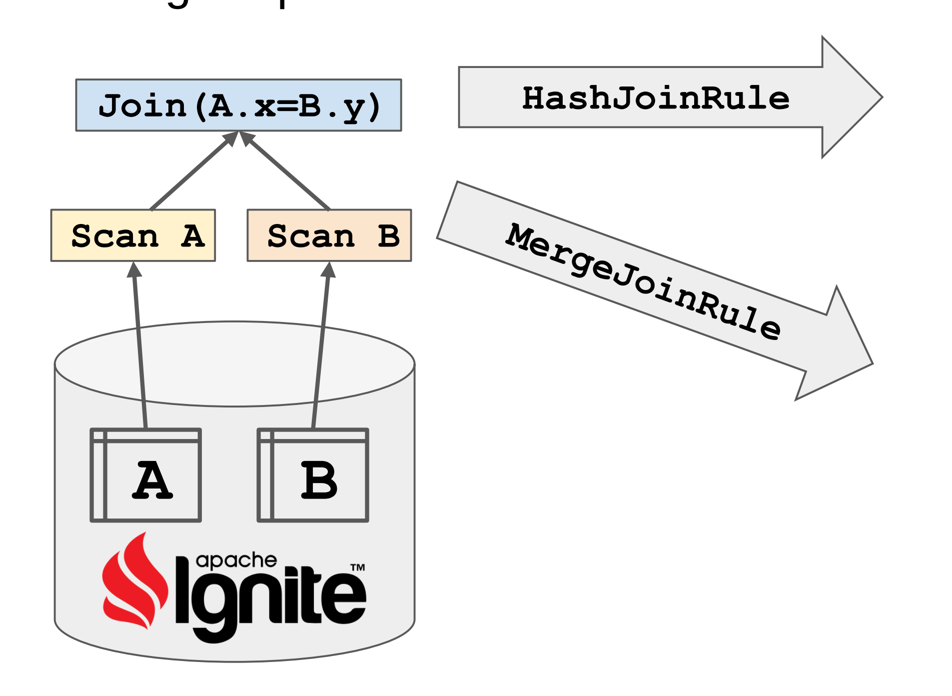


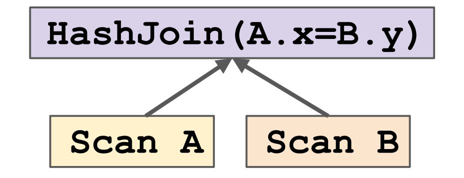


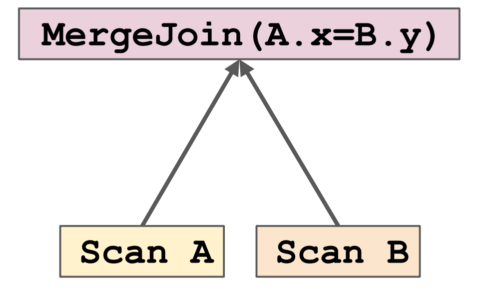
VolcanoPlanner – физическое планирование

Logical plan

Physical plan

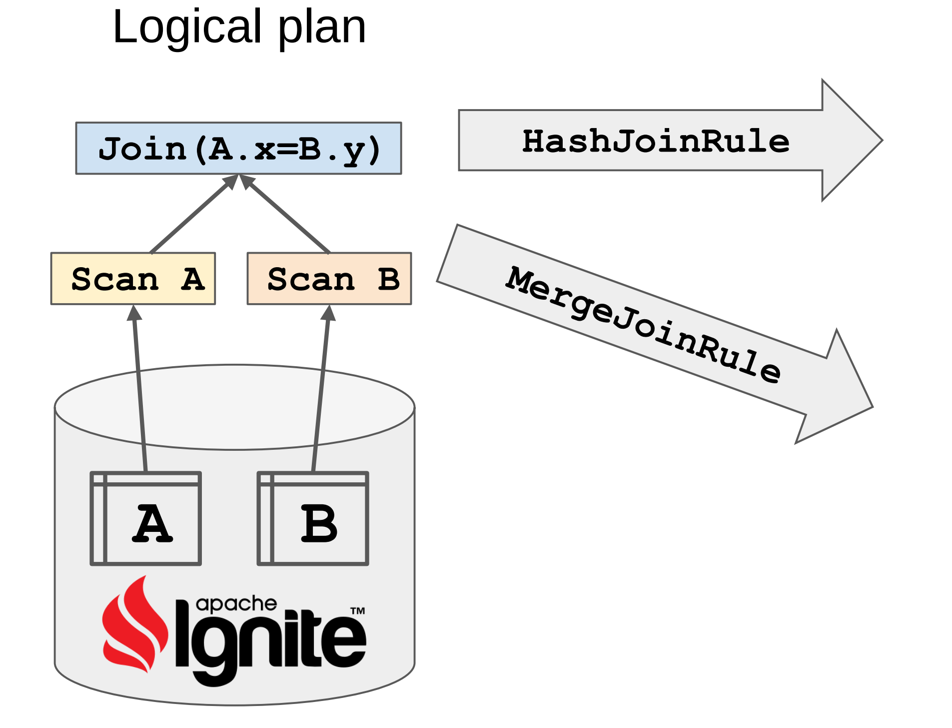


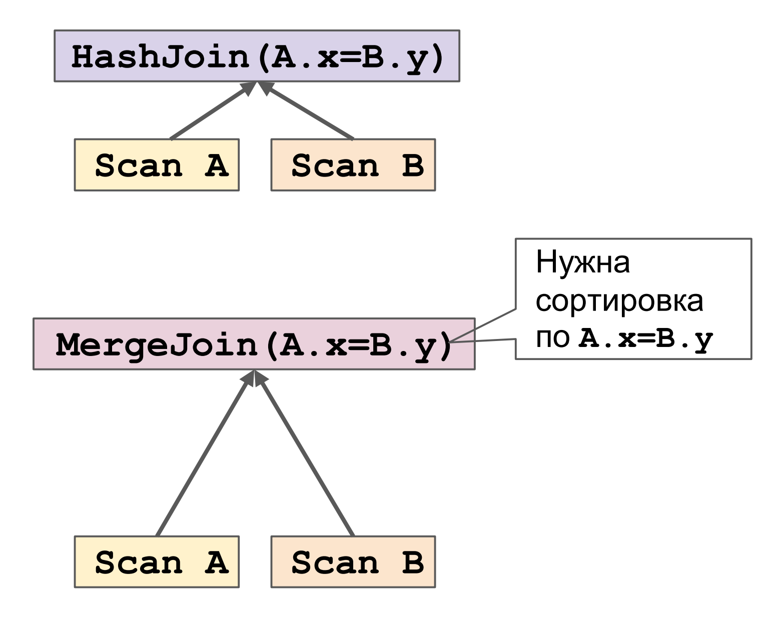






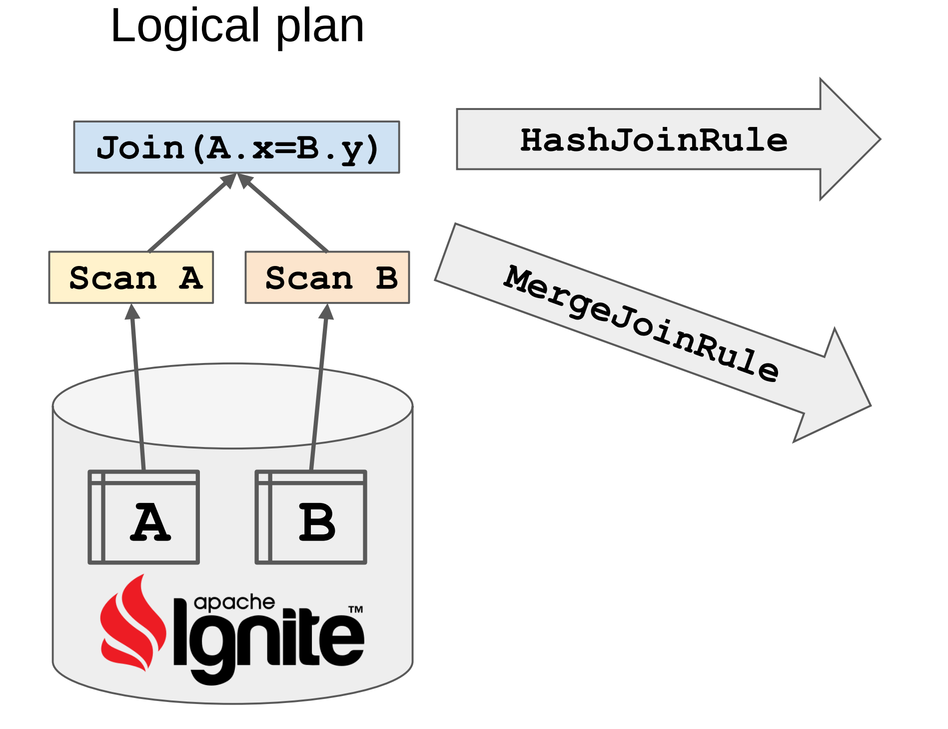
VolcanoPlanner – физическое планирование

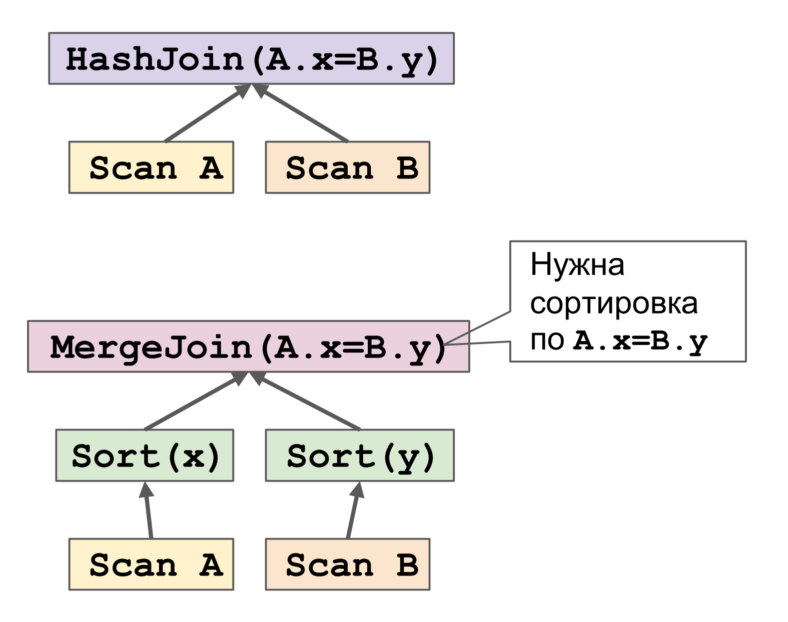






VolcanoPlanner – физическое планирование







VolcanoPlanner – физическое планирование

Logical plan

Physical plan

Join(A.x=B.y)

Scan A

Scan B

A

B

apache Ignite™

HashJoinRule

MergeJoinRule

HashJoin(A.x=B.y)

10

Scan A

Scan B

MergeJoin(A.x=B.y)

Нужна сортировка по **A.x=B.y**

Sort(x)

Sort(y)

Scan A

Scan B



VolcanoPlanner – физическое планирование

Logical plan

Physical plan

Join(A.x=B.y)

Scan A

Scan B

A

B

apache Ignite™

HashJoinRule

MergeJoinRule

HashJoin(A.x=B.y)

10

Scan A

Scan B

20

MergeJoin(A.x=B.y)

Нужна сортировка по **A.x=B.y**

Sort(x)

Sort(y)

Scan A

Scan B



VolcanoPlanner – физическое планирование

Logical plan

Physical plan

Join(A.x=B.y)

Scan A

Scan B

A

B

apache Ignite™

HashJoinRule

MergeJoinRule

60 HashJoin(A.x=B.y)

10 Scan A

Scan B 20

Нужна сортировка по **A.x=B.y**

MergeJoin(A.x=B.y)

Sort(x)

Sort(y)

Scan A

Scan B



VolcanoPlanner – физическое планирование

Logical plan

Physical plan

Join(A.x=B.y)

Scan A

Scan B

A

B

apache Ignite™

HashJoinRule

MergeJoinRule

60 HashJoin(A.x=B.y)

90

10 Scan A

Scan B 20

MergeJoin(A.x=B.y)

Нужна сортировка по A.x=B.y

Sort(x)

Sort(y)

Scan A

Scan B



VolcanoPlanner – физическое планирование

Logical plan

Physical plan



VolcanoPlanner – физическое планирование

Logical plan

Physical plan



VolcanoPlanner – физическое планирование

Logical plan

Physical plan



VolcanoPlanner – физическое планирование

Logical plan

Physical plan



VolcanoPlanner – физическое планирование

Logical plan

Join(A.x=B.y)

Scan A

Scan B

A

B

apache Ignite™

HashJoinRule

MergeJoinRule

Physical plan

60 HashJoin(A.x=B.y)

90

10 Scan A

Scan B 20

30 MergeJoin(A.x=B.y)

Нужна сортировка по **A.x=B.y**

33 Sort(x)

Sort(y) 45

10 Scan A

Scan B 20



VolcanoPlanner – физическое планирование

Logical plan

Physical plan

Join(A.x=B.y)

Scan A

Scan B

A

B

apache Ignite™

HashJoinRule

MergeJoinRule

60 HashJoin(A.x=B.y)

90

10 Scan A

Scan B 20

30 MergeJoin(A.x=B.y)

Нужна сортировка по **A.x=B.y**

33 Sort(x)

Sort(y) 45

138

10 Scan A

Scan B 20



VolcanoPlanner – физическое планирование

Logical plan

Physical plan

Join(A.x=B.y)

Scan A

Scan B

A

B

apache Ignite™

HashJoinRule

MergeJoinRule

WINNER!

60 HashJoin(A.x=B.y)

90

10 Scan A

Scan B 20

Нужна сортировка по A.x=B.y

30 MergeJoin(A.x=B.y)

33 Sort(x)

Sort(y) 45

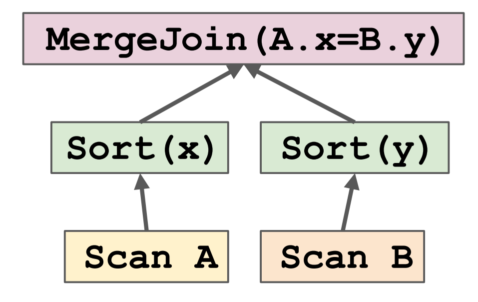
138

10 Scan A

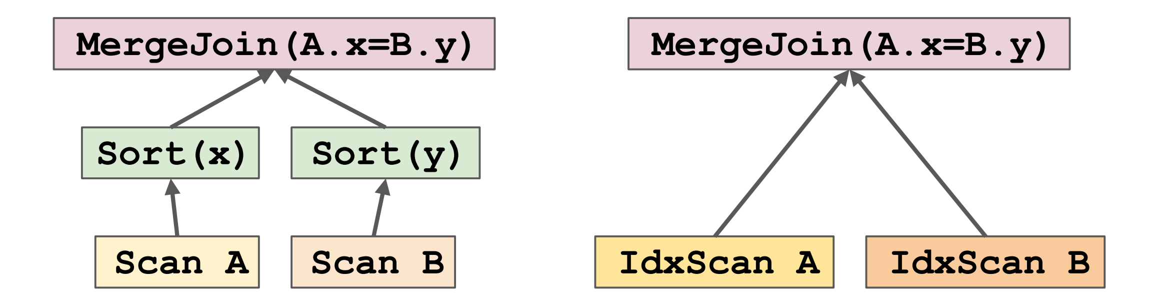
Scan B 20



Сортировка и причем тут трейты

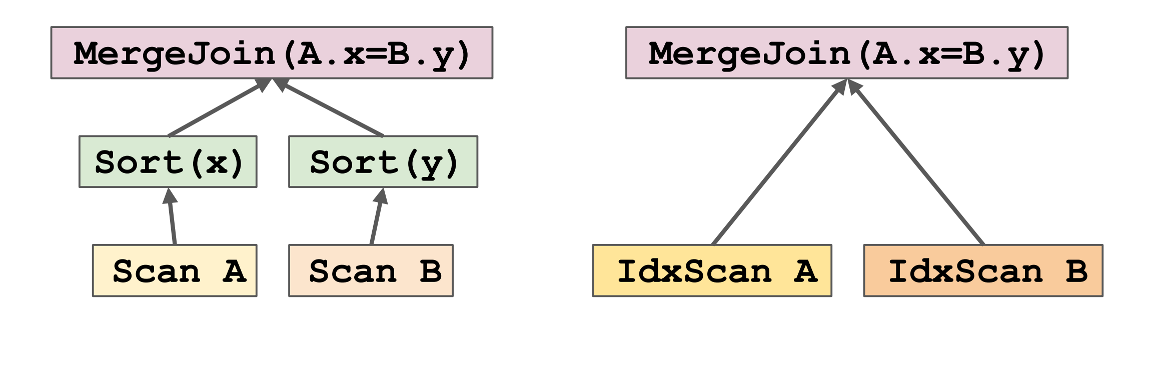


Сортировка и причем тут трейты





Сортировка и причем тут трейты





Сортировка и причем тут трейты

MergeJoin(A.x=B.y)

Sort(x)

Sort(y)

Scan A

Scan B

MergeJoin(A.x=B.y)

IdxScan A

IdxScan B

MergeJoin(A.x=B.y)

Sorted by [x]



Сортировка и причем тут трейты

MergeJoin(A.x=B.y)

Sort(x) Sort(y)

Scan A Scan B

MergeJoin(A.x=B.y)

IdxScan A IdxScan B

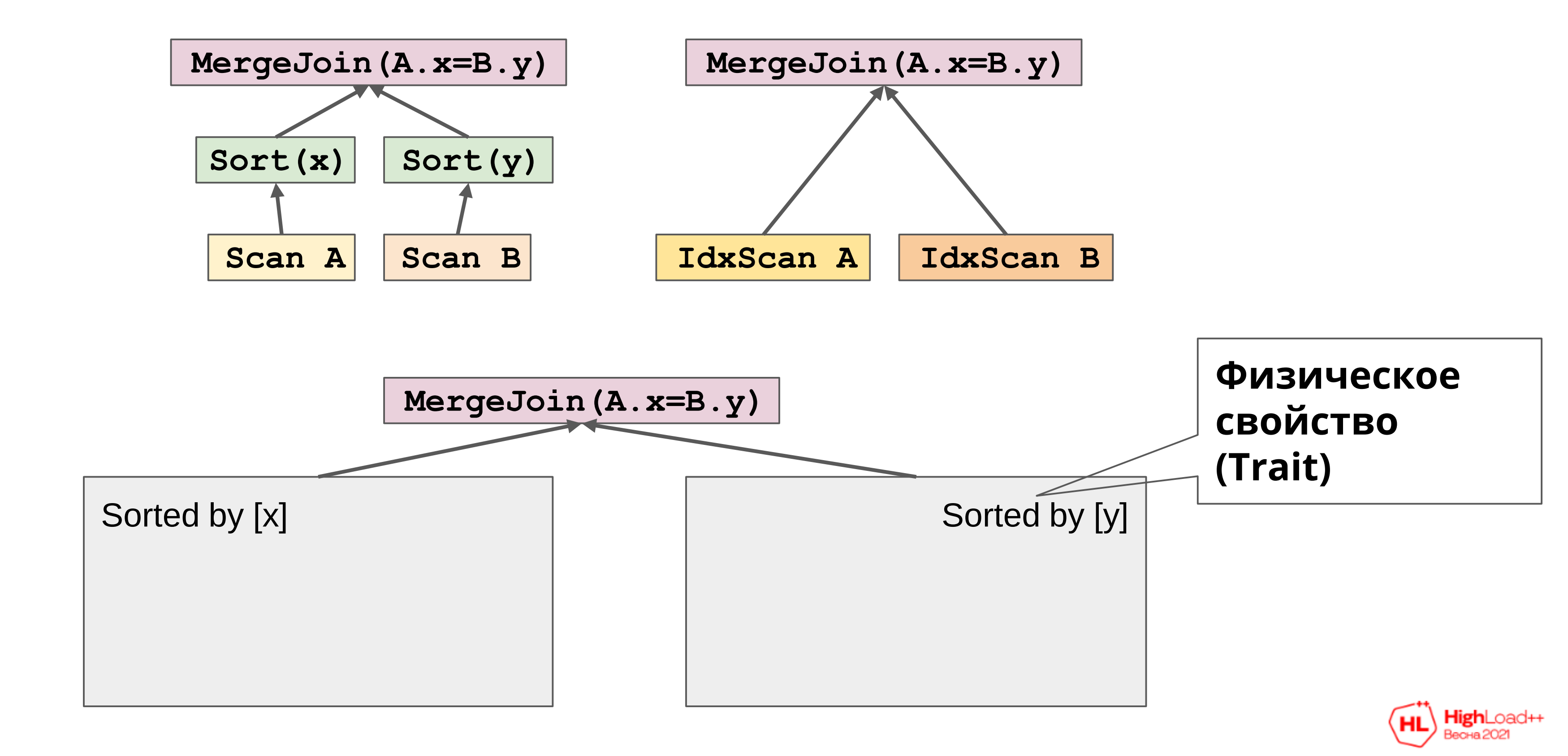
MergeJoin(A.x=B.y)

Sorted by [x]

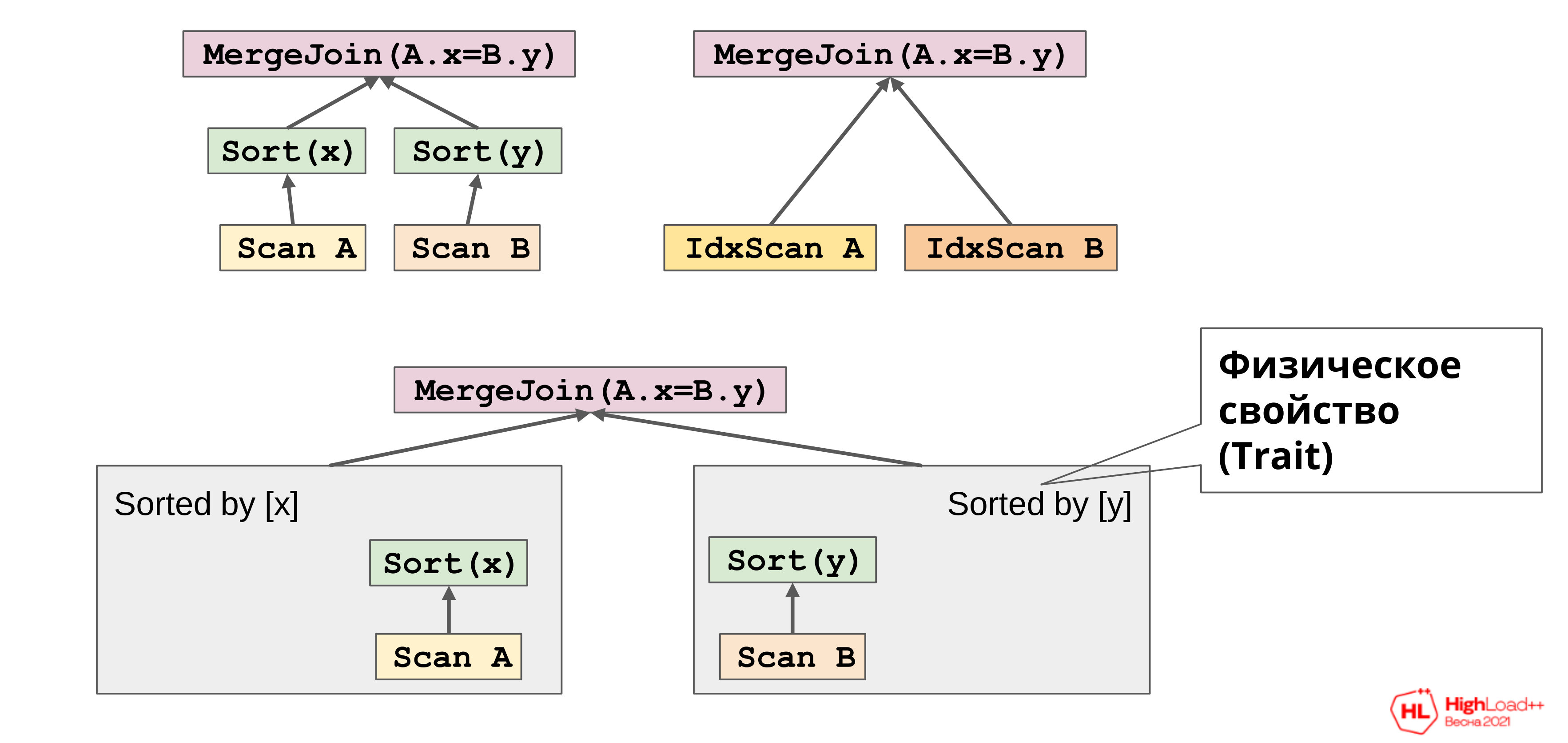
Sorted by [y]



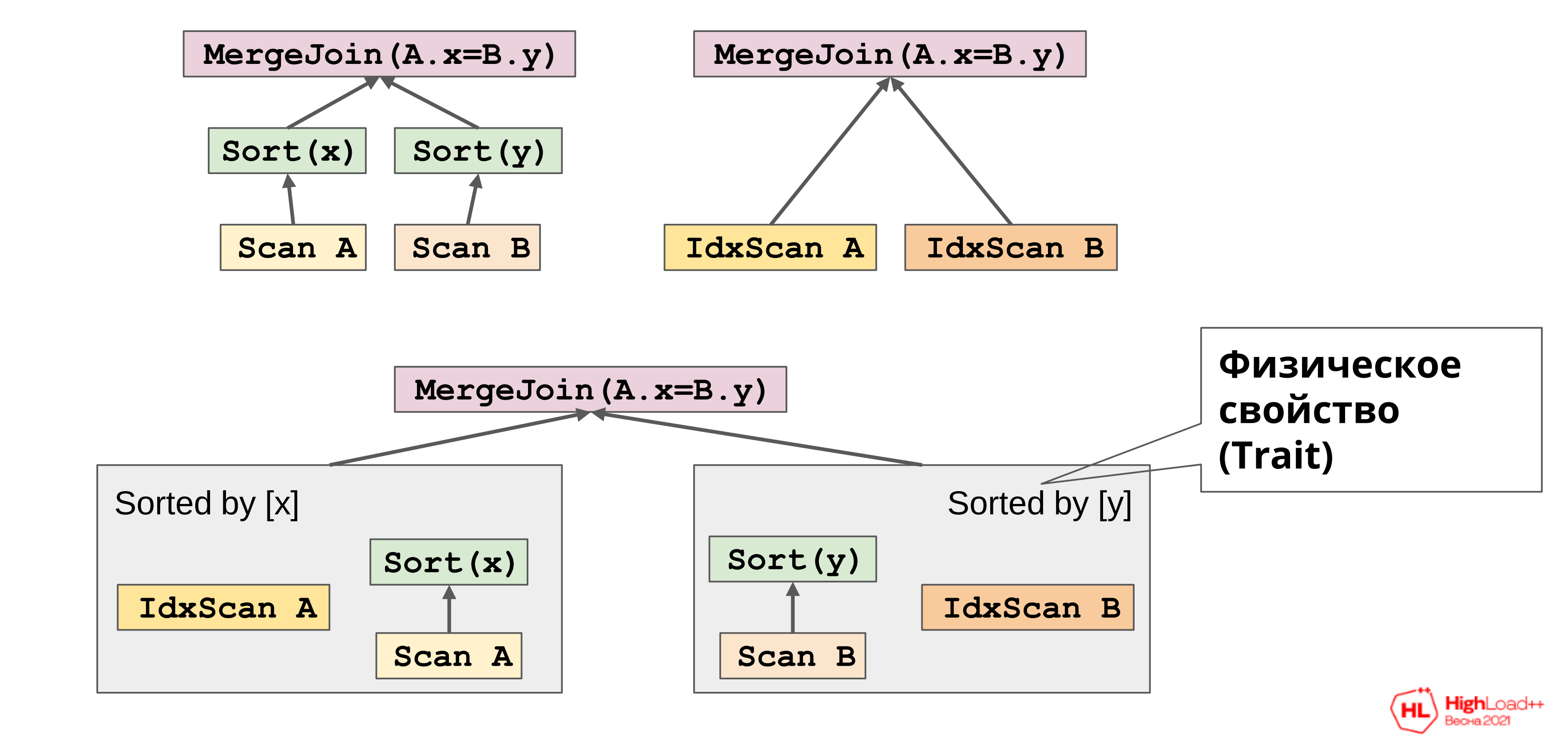
Сортировка и причем тут трейты



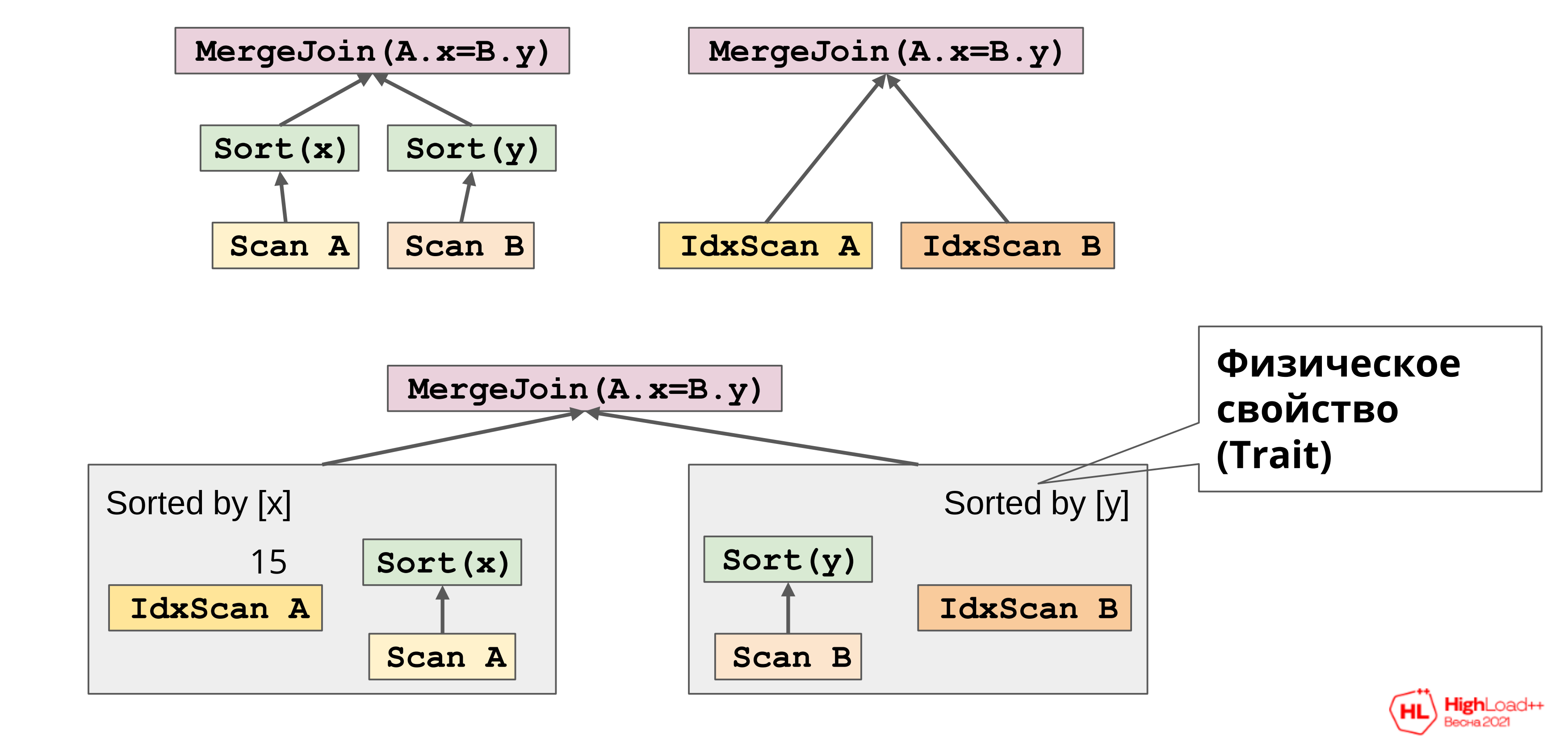
Сортировка и причем тут трейты



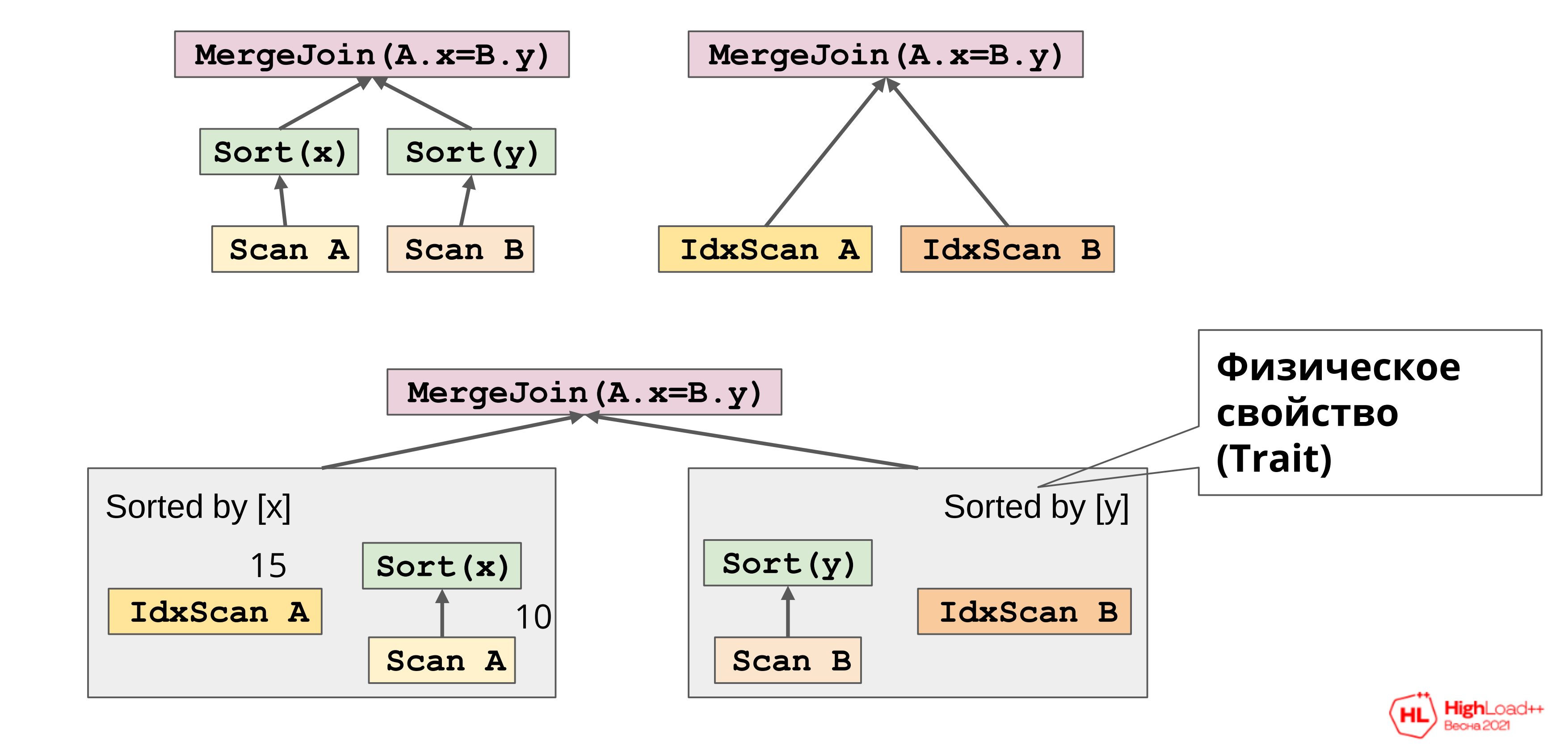
Сортировка и причем тут трейты



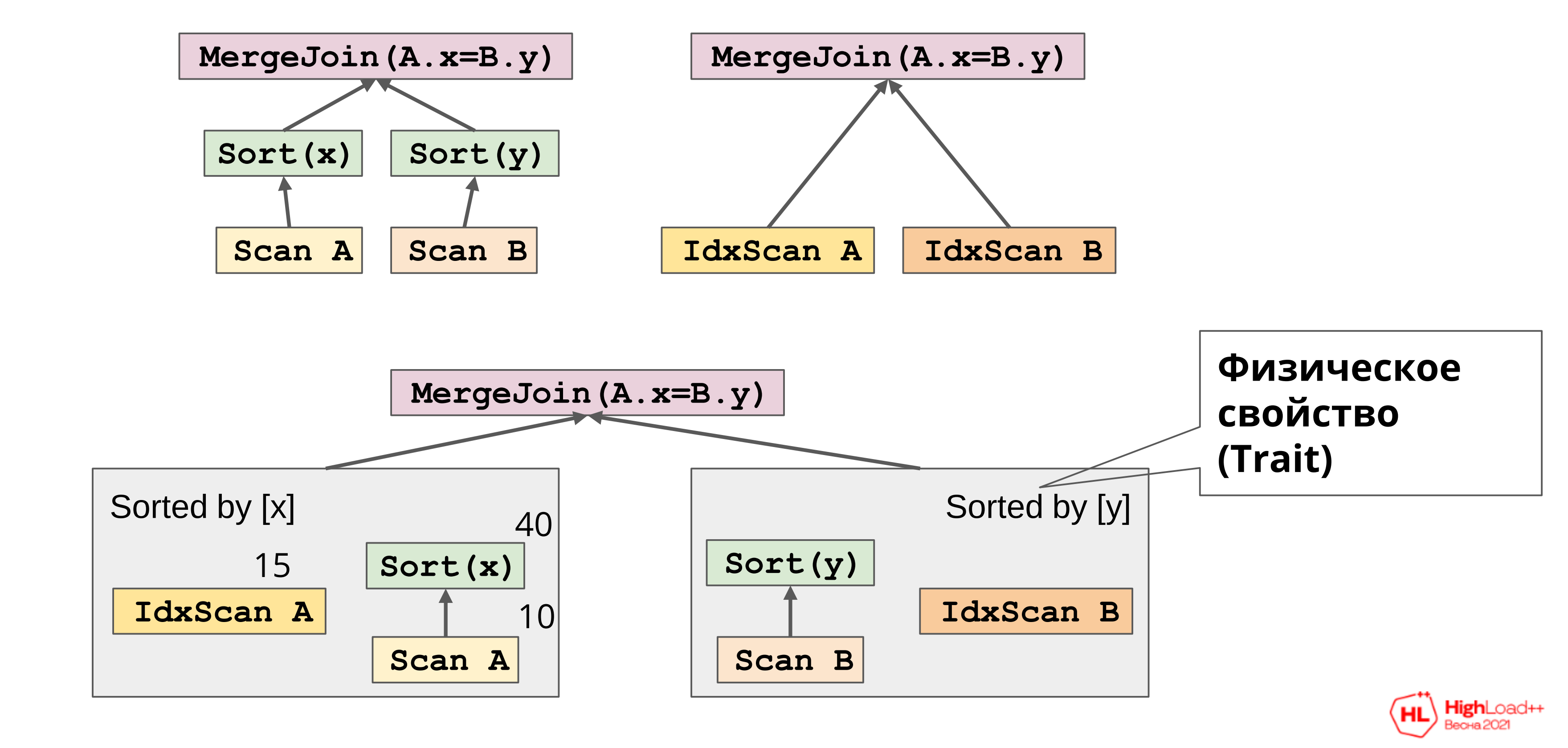
Сортировка и причем тут трейты



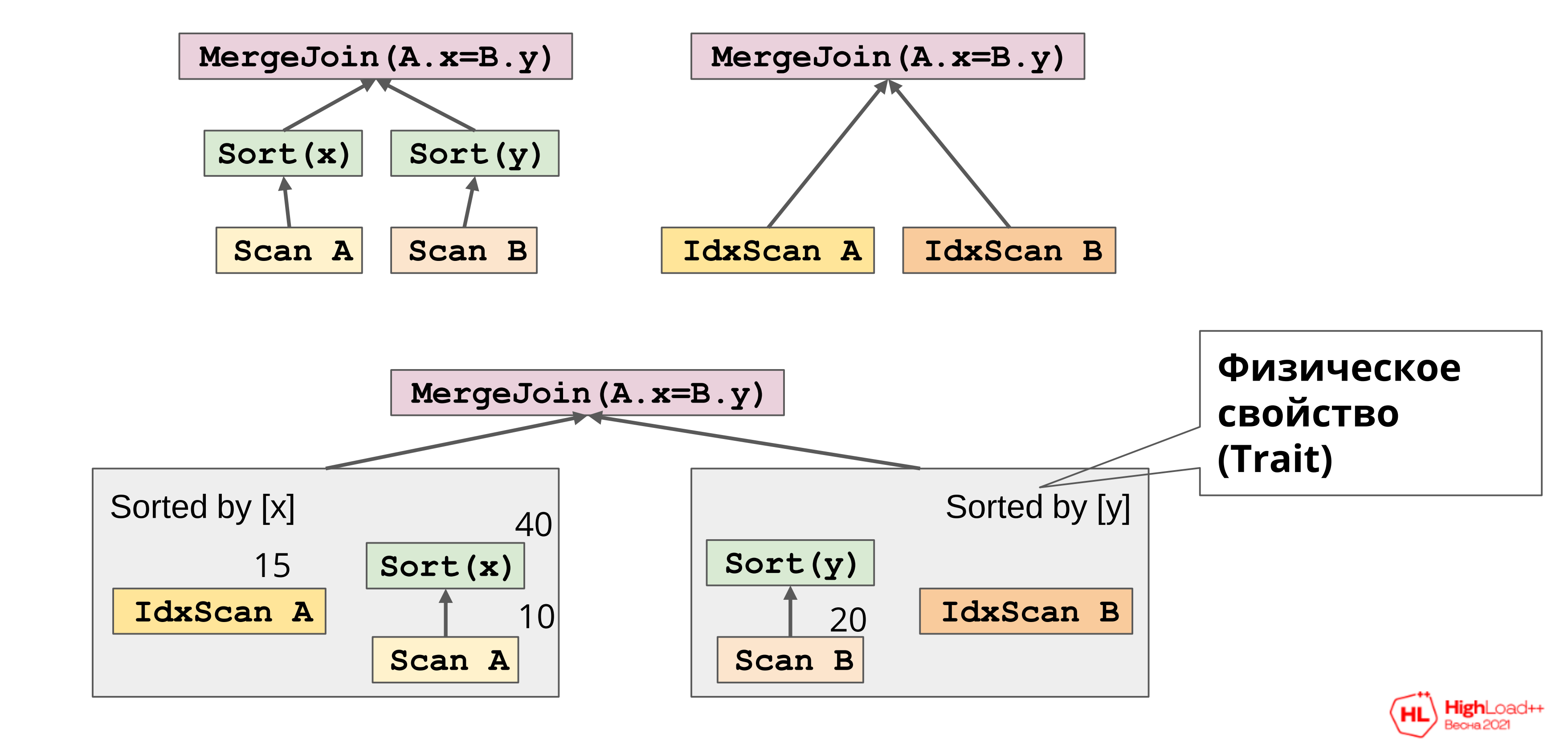
Сортировка и причем тут трейты



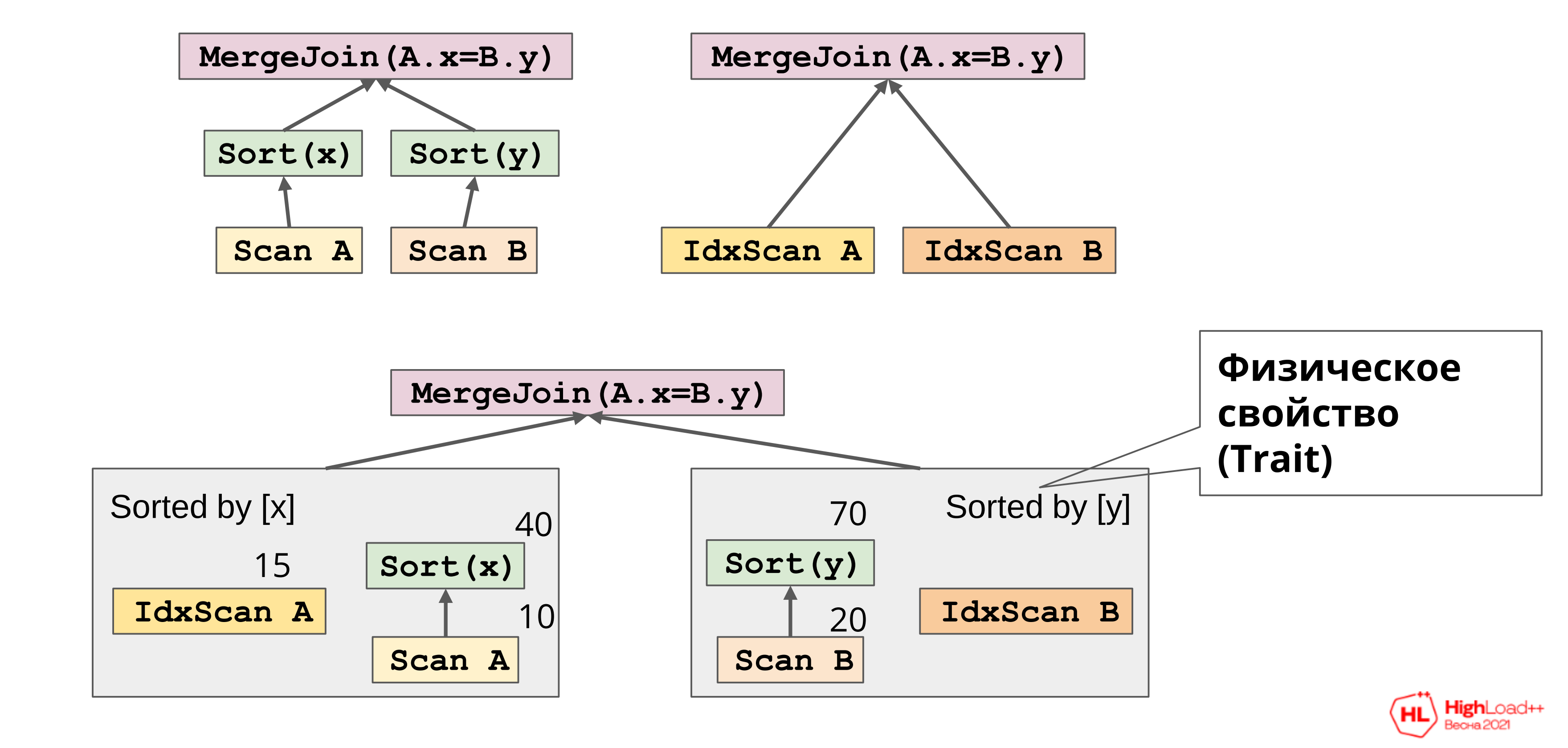
Сортировка и причем тут трейты



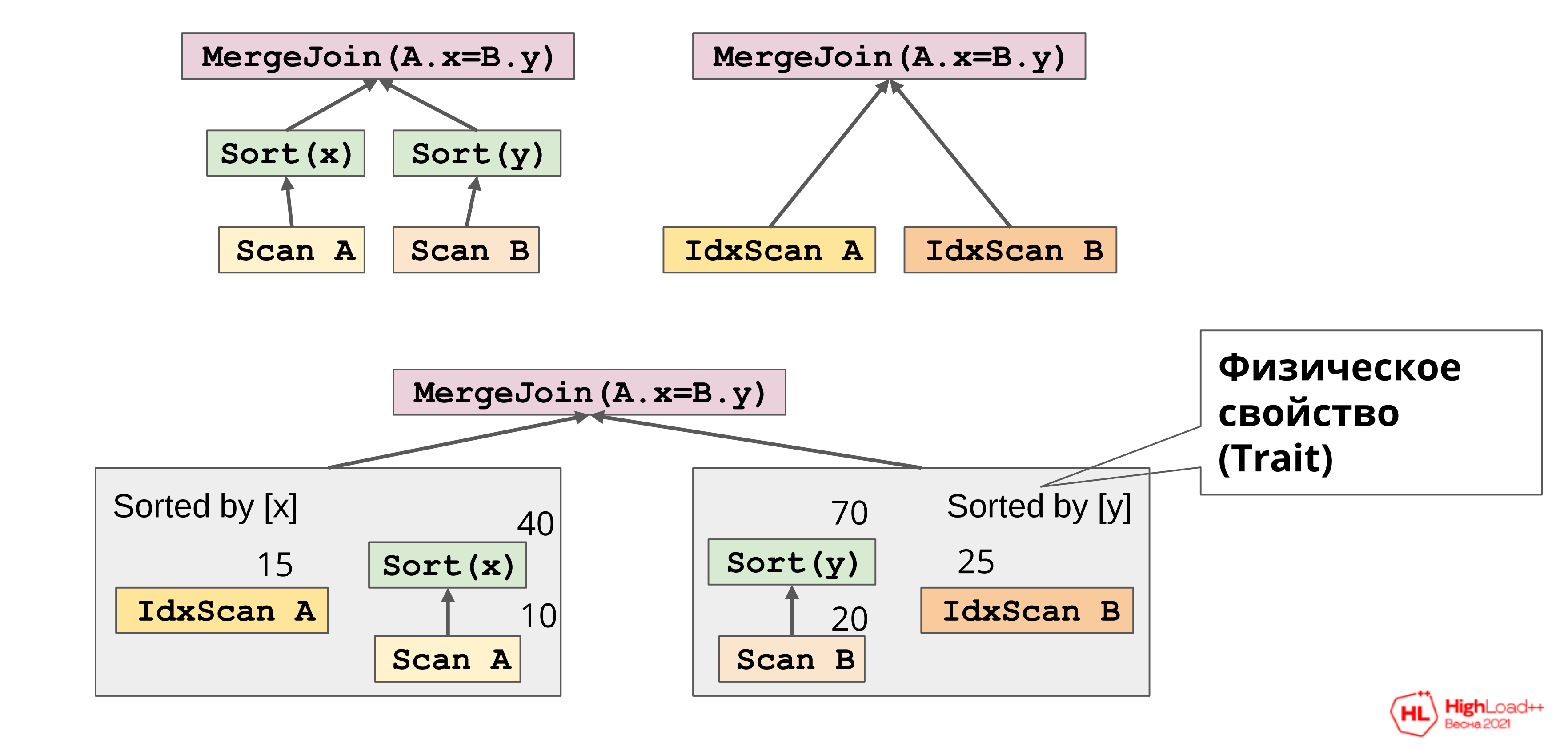
Сортировка и причем тут трейты



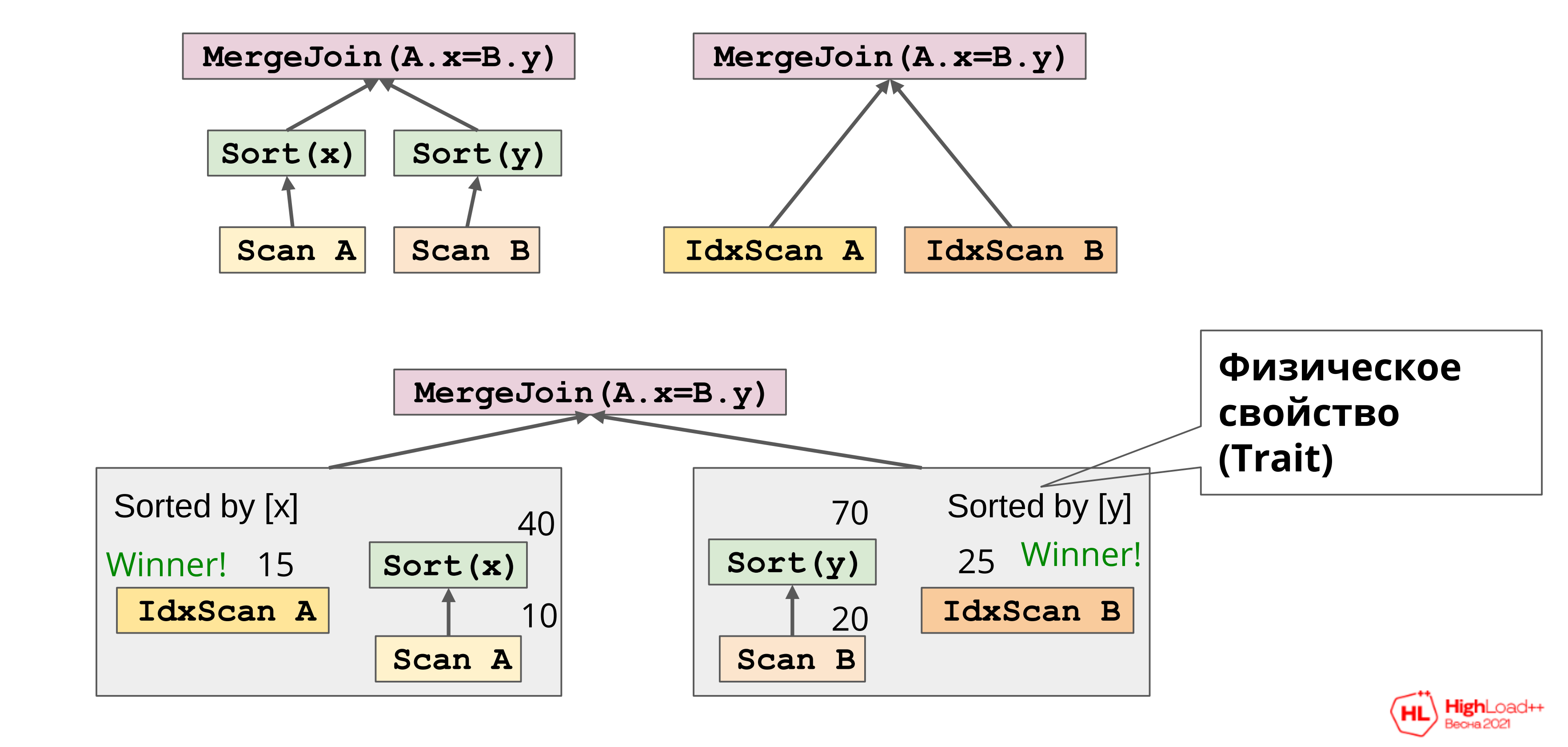
Сортировка и причем тут трейты



Сортировка и причем тут трейты



Сортировка и причем тут трейты



Что делать с планом?

PHASE 1

PHASE 2

PHASE 3

Получаем план из Calcite

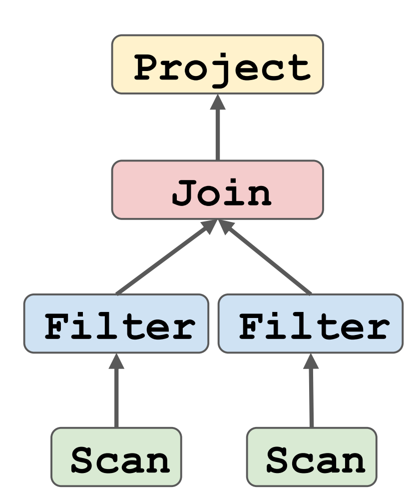
?

Отдаем юзеру результат запроса

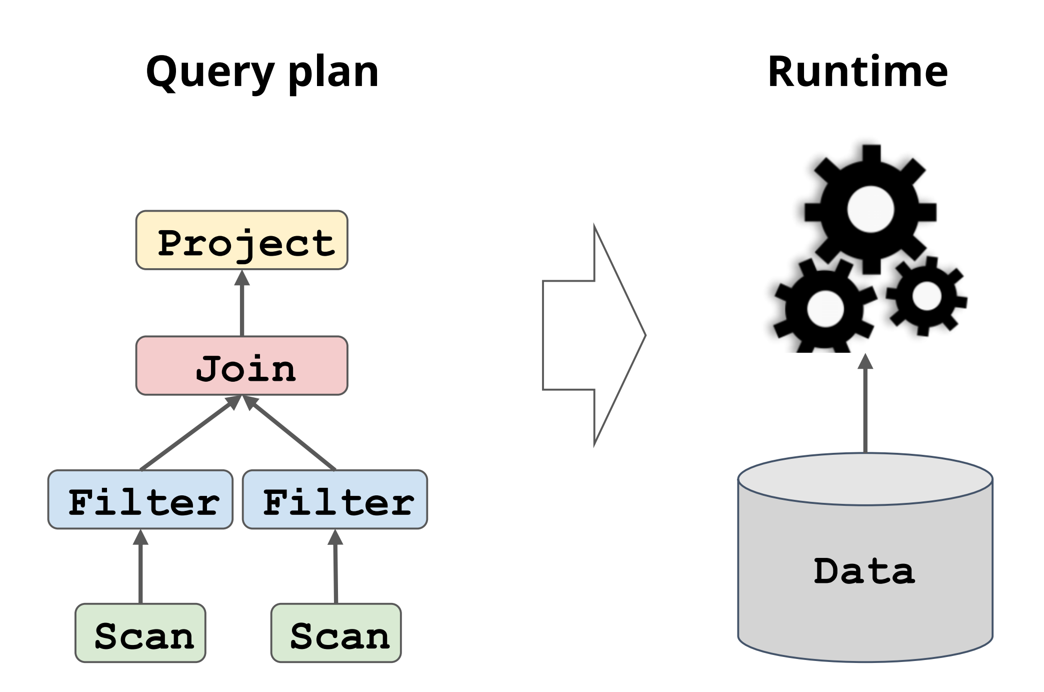


Рантайм?

Query plan



Рантайм?


Query plan
Runtime
Project
Join
Filter
Filter
Scan
Scan
Data


HL
HighLoad++
Весна 2021

Рантайм?

Query plan | **Runtime** | **Query result**

Project
Join
Filter Filter
Scan Scan

Data

1	a
2	b
3	c
4	d
5	e



И как сделать рантайм?

```
SELECT name FROM emps WHERE id = 10
```



И как сделать рантайм?

```
SELECT name FROM emps WHERE id = 10
```





И как сделать рантайм?

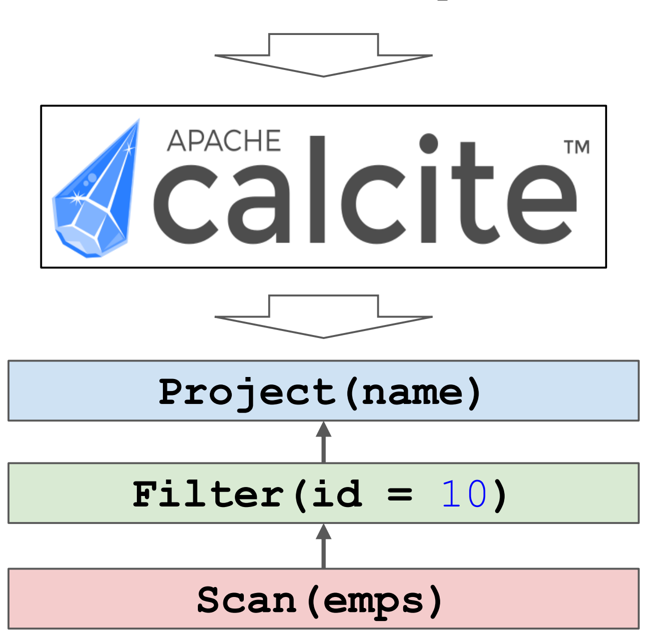
```
SELECT name FROM emps WHERE id = 10
```





И как сделать рантайм?

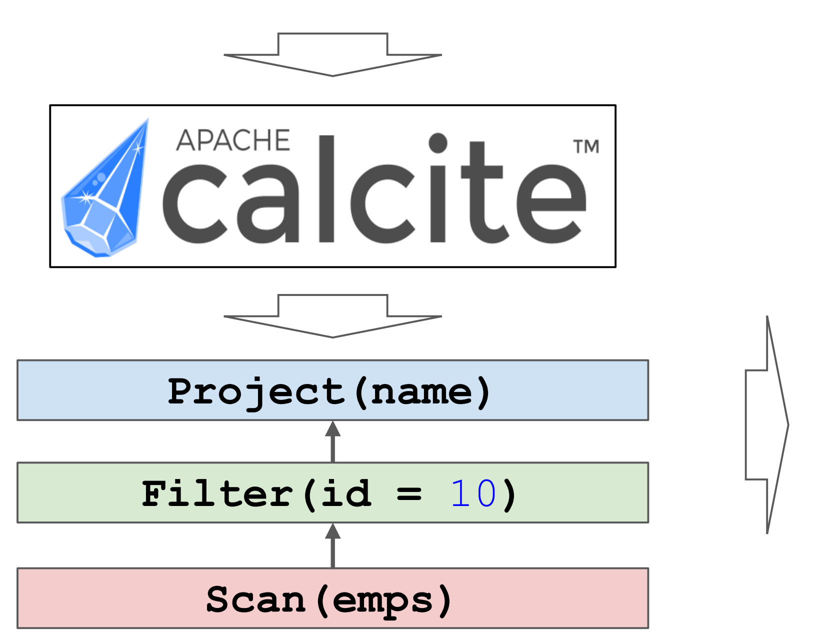
```
SELECT name FROM emps WHERE id = 10
```





И как сделать рантайм?

```
SELECT name FROM emps WHERE id = 10
```

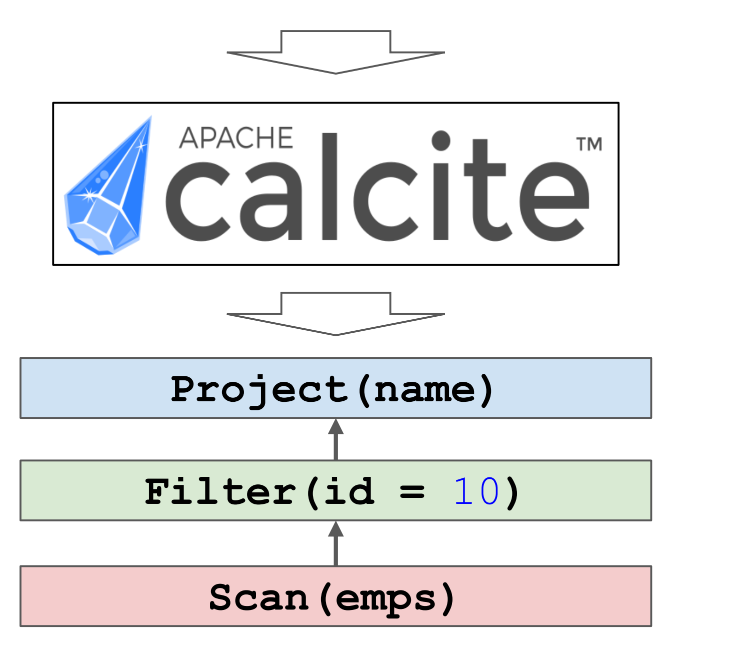



```
for row : emps
  emit(row)
```



И как сделать рантайм?

```
SELECT name FROM emps WHERE id = 10
```



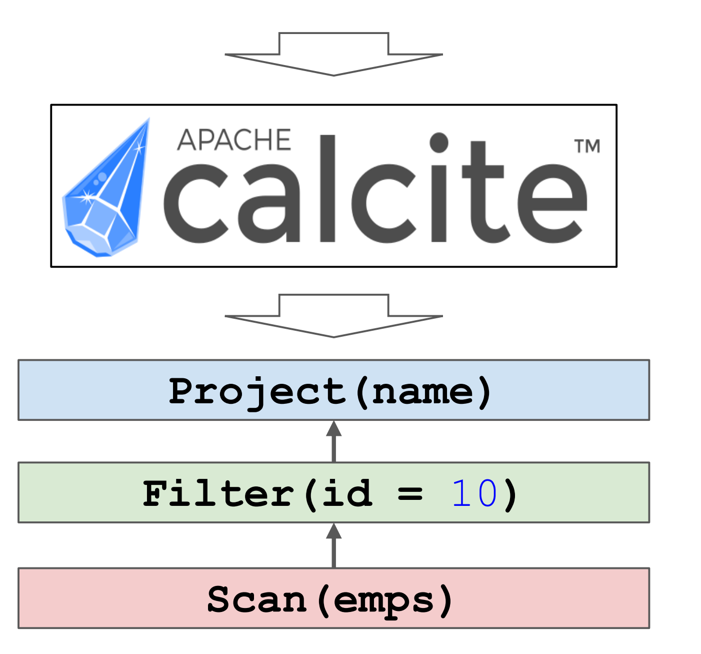

```
for row : child.next()
  if row[id] = 10
    emit(row)
```

```
for row : emps
  emit(row)
```



И как сделать рантайм?

```
SELECT name FROM emps WHERE id = 10
```



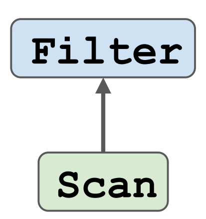

```
for row : child.next()
 emit(row[name]))
```

```
for row : child.next()
  if row[id] = 10
    emit(row)
```

```
for row : emps
  emit(row)
```

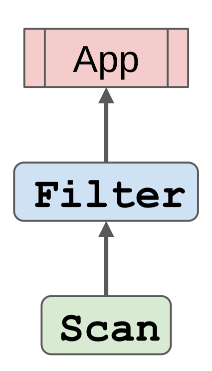


А как быть в распределенной системе?





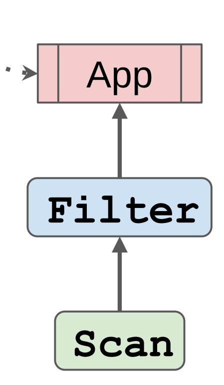
А как быть в распределенной системе?





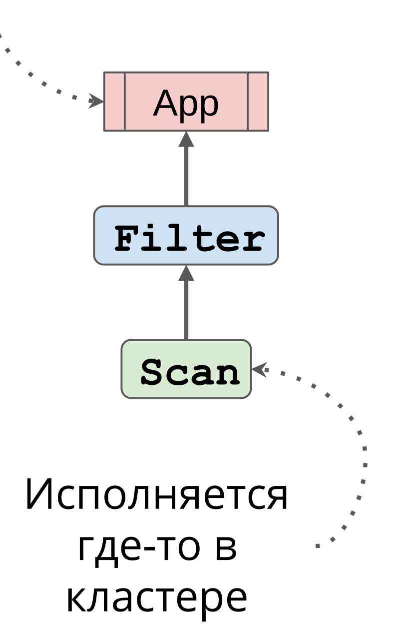
А как быть в распределенной системе?

Исполняется на клиентском узле





А как быть в распределенной системе?





А как быть в распределенной системе?

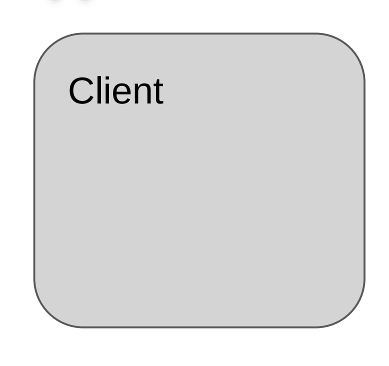
Исполняется на клиентском узле

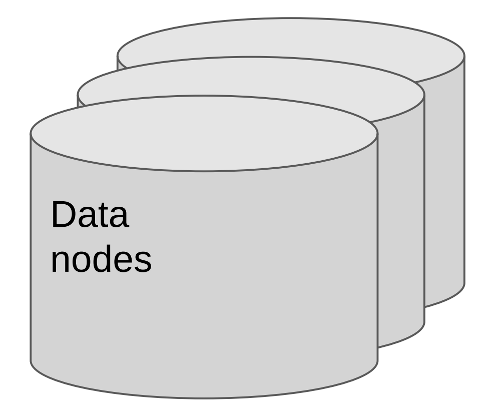
App

Filter

Scan

Исполняется где-то в кластере







А как быть в распределенной системе?

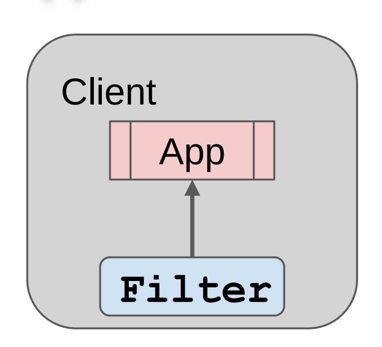
Исполняется на клиентском узле

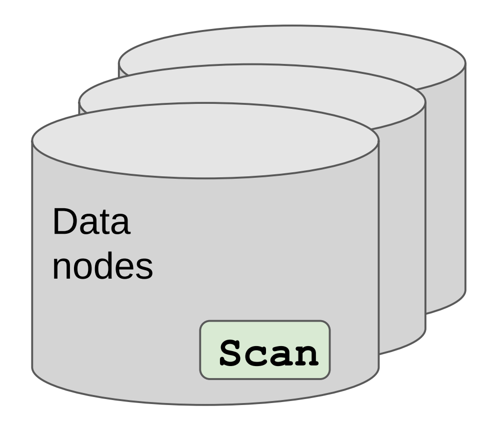
App

Filter

Scan

Исполняется где-то в кластере







А как быть в распределенной системе?

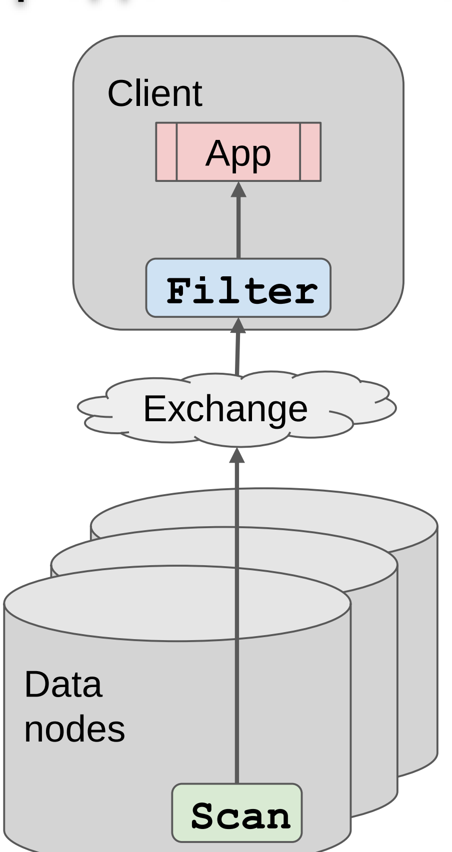
Исполняется на клиентском узле

App

Filter

Scan

Исполняется где-то в кластере





А как быть в распределенной системе?

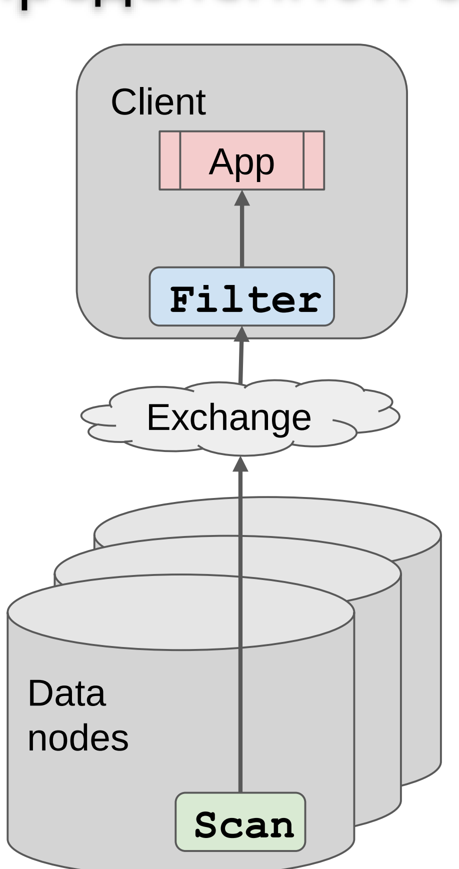
Исполняется на клиентском узле

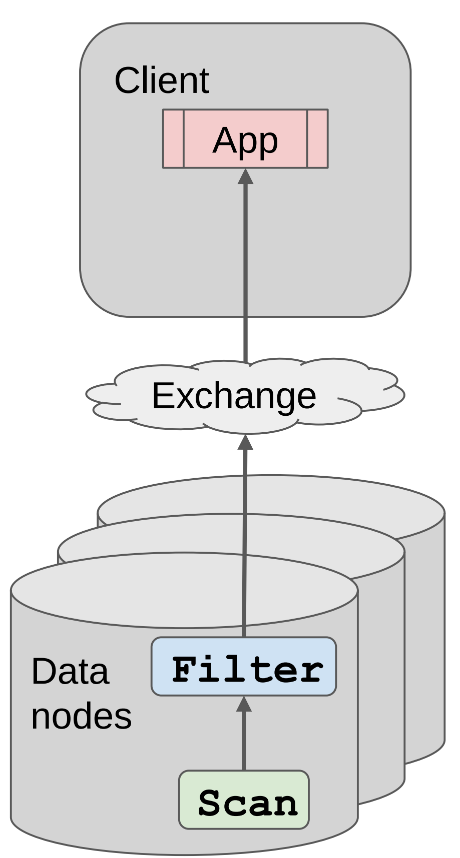
App

Filter

Scan

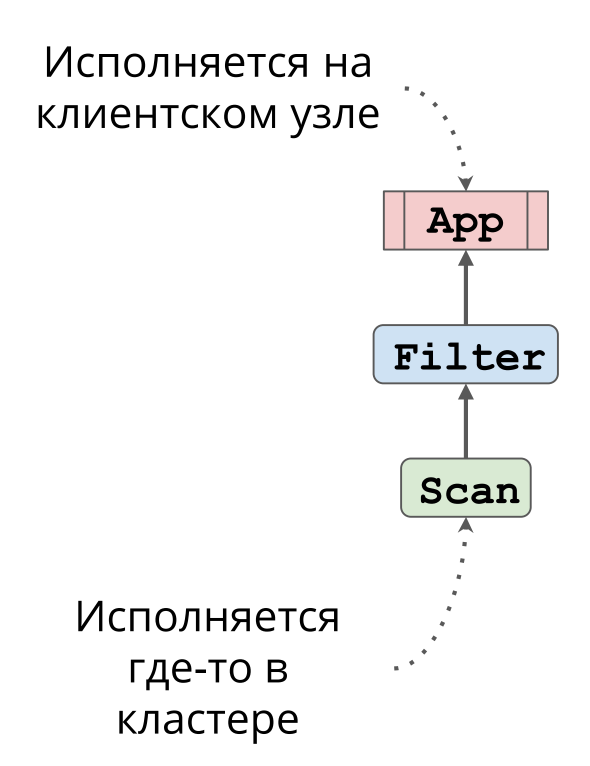
Исполняется где-то в кластере





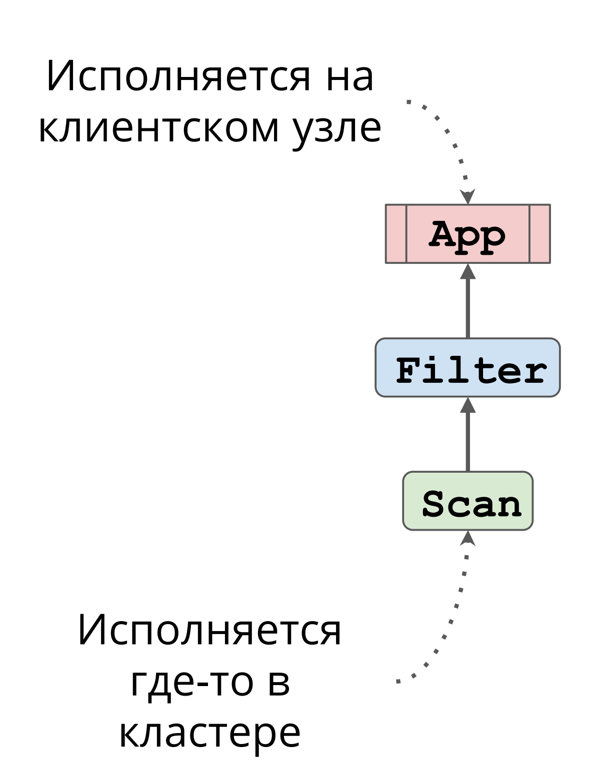


Как сказать кальциту о распределенных данных?





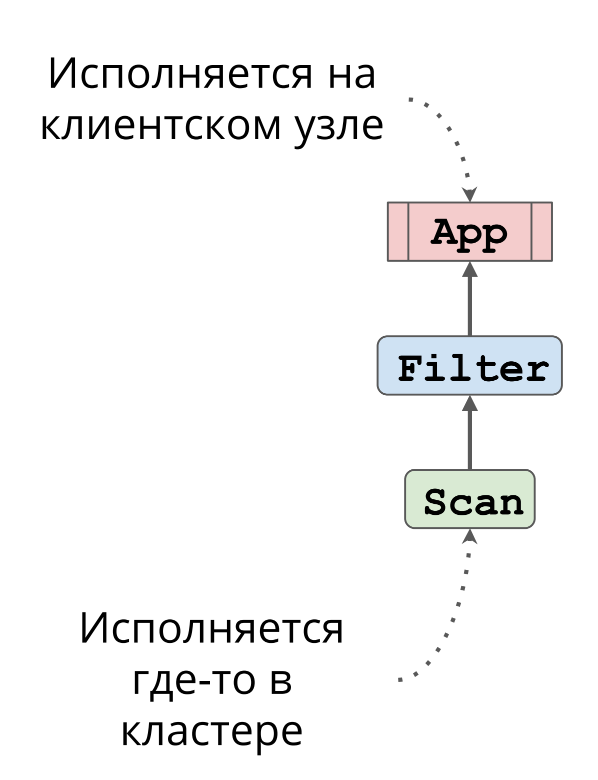
Как сказать кальциту о распределенных данных?



Трейт распределения в кластере
(как сортировка, только распределение)



Как сказать кальциту о распределенных данных?



Трейт распределения в кластере
(как сортировка, только распределение)

App [Single]



Как сказать кальциту о распределенных данных?

Исполняется на клиентском узле

App

Filter

Scan

Исполняется где-то в кластере

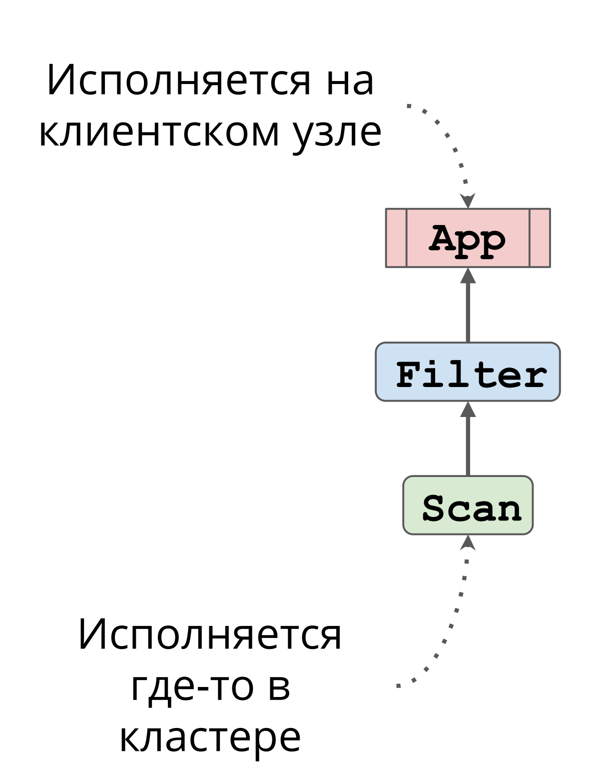
Трейт распределения в кластере
(как сортировка, только распределение)

App [Single]

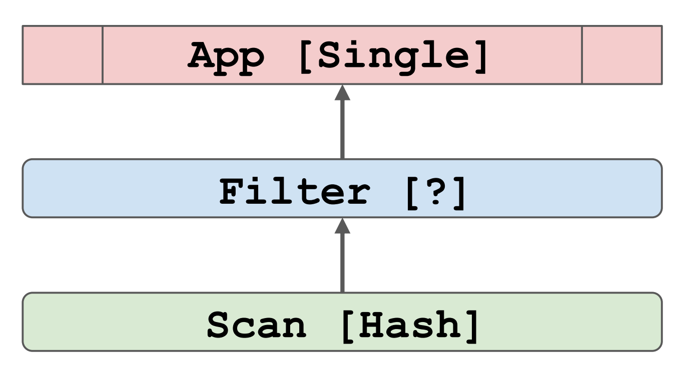
Scan [Hash]



Как сказать кальциту о распределенных данных?

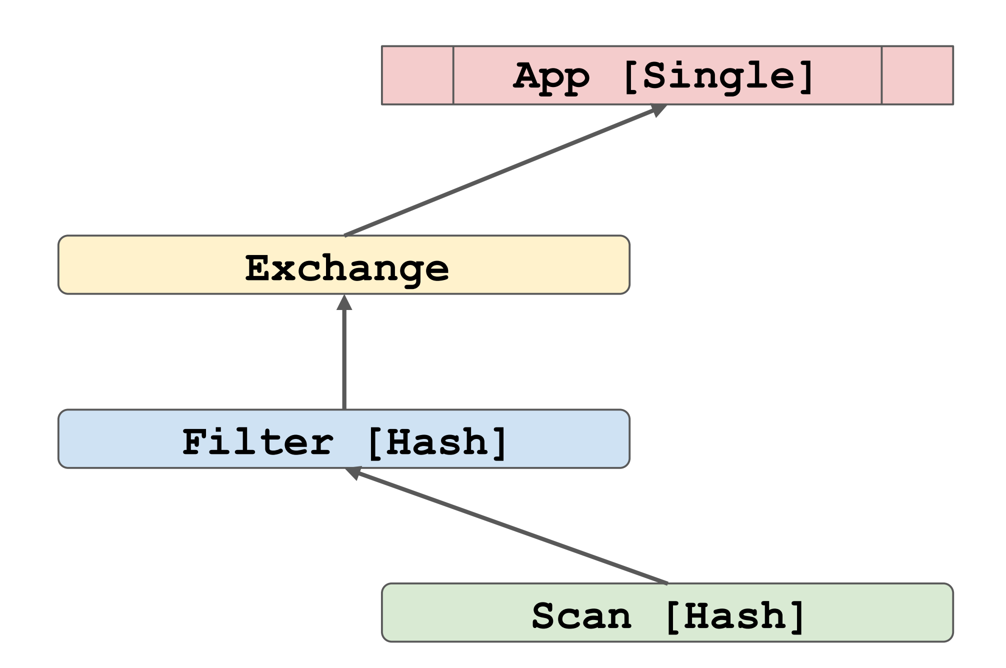


Трейт распределения в кластере
(как сортировка, только распределение)





Оптимизатор и трейты





Оптимизатор и трейты

App [Single]

Exchange

Filter [Single]

Filter [Hash]

Exchange

Scan [Hash]



Оптимизатор и трейты

App [Single]

Exchange

Filter [Single]

Filter [Hash]

Exchange

100

Scan [Hash]



Оптимизатор и трейты

App [Single]

Exchange

Filter [Single]

10

Filter [Hash]

Exchange

100

Scan [Hash]



Оптимизатор и трейты

App [Single]

10

Exchange

Filter [Single]

10

Filter [Hash]

Exchange

100

Scan [Hash]



Оптимизатор и трейты

App [Single]

10

Exchange

Filter [Single]

10

Filter [Hash]

Exchange

100

100

Scan [Hash]



Оптимизатор и трейты

App [Single]

10

Exchange

Filter [Single]

10

100

Filter [Hash]

Exchange

100

100

Scan [Hash]



Оптимизатор и трейты

App [Single]

10

10

Exchange

Filter [Single]

10

100

Filter [Hash]

Exchange

100

100

Scan [Hash]



Оптимизатор и трейты

Winner!!!

App [Single]

10

10

Exchange

Filter [Single]

10

100

Filter [Hash]

Exchange

100

100

Scan [Hash]



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
 Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
SingletonExchange // Отправляем локальные агрегаты на клиента
   Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
    Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
Agg(#0=[SUM(#0)], #1=[SUM(#1)]) // Считаем глобальные агрегаты
  SingletonExchange // Отправляем локальные агрегаты на клиента
    Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
     Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
Project(#0=[#0 / #1]) // Получаем среднее число как сумма/количество
  Agg(#0=[SUM(#0)], #1=[SUM(#1)]) // Считаем глобальные агрегаты
    SingletonExchange // Отправляем локальные агрегаты на клиента
      Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
       Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
BroadcastExchange // Отправляем полученное среднее на все ноды
  Project(#0=[#0 / #1]) // Получаем среднее число как сумма/количество
    Agg(#0=[SUM(#0)], #1=[SUM(#1)]) // Считаем глобальные агрегаты
      SingletonExchange // Отправляем локальные агрегаты на клиента
        Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
         Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
Scan(table=[emps]) // Снова сканируем emps на всех нодах
BroadcastExchange // Отправляем полученное среднее на все ноды
  Project(#0=[#0 / #1]) // Получаем среднее число как сумма/количество
    Agg(#0=[SUM(#0)], #1=[SUM(#1)]) // Считаем глобальные агрегаты
      SingletonExchange // Отправляем локальные агрегаты на клиента
        Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
         Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
Scan(table=[emps]) // Снова сканируем emps на всех нодах
BroadcastExchange // Отправляем полученное среднее на все ноды
  Project(#0=[#0 / #1]) // Получаем среднее число как сумма/количество
    Agg(#0=[SUM(#0)], #1=[SUM(#1)]) // Считаем глобальные агрегаты
      SingletonExchange // Отправляем локальные агрегаты на клиента
        Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
         Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
HashJoin(condition=[salary=#0]) // Джойним emps со средним
  Scan(table=[emps]) // Снова сканируем emps на всех нодах
  BroadcastExchange // Отправляем полученное среднее на все ноды
    Project(#0=[#0 / #1]) // Получаем среднее число как сумма/количество
      Agg(#0=[SUM(#0)], #1=[SUM(#1)]) // Считаем глобальные агрегаты
        SingletonExchange // Отправляем локальные агрегаты на клиента
          Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
           Scan(table=[emps]) // Сканируем emps на всех нодах
```



Apache Ignite смог!

```
SELECT * FROM emps WHERE emps.salary =
(SELECT AVG(emps.salary) FROM emps)
```

APACHE calcite™

```
SingletonExchange // Отправляем полученный результат на клиента
  HashJoin(condition=[salary=#0]) // Джойним emps со средним
    Scan(table=[emps]) // Снова сканируем emps на всех нодах
    BroadcastExchange // Отправляем полученное среднее на все ноды
      Project(#0=[#0 / #1]) // Получаем среднее число как сумма/количество
        Agg(#0=[SUM(#0)], #1=[SUM(#1)]) // Считаем глобальные агрегаты
          SingletonExchange // Отправляем локальные агрегаты на клиента
            Agg(#0=[SUM(salary)], #1=[COUNT(1)]) // Считаем локальные агрегаты
             Scan(table=[emps]) // Сканируем emps на всех нодах
```



А можно кастомизировать Calcite?



А можно кастомизировать Calcite?

- Добавить свои операторы RelNode



А можно кастомизировать Calcite?

- Добавить свои операторы RelNode
- Добавить свои правила оптимизатора



А можно кастомизировать Calcite?

- Добавить свои операторы RelNode
- Добавить свои правила оптимизатора
- Переопределить косты у своих и встроенных операторов



А можно кастомизировать Calcite?

- Добавить свои операторы RelNode
- Добавить свои правила оптимизатора
- Переопределить косты у своих и встроенных операторов
- Добавить свои метаданные и использовать их при планировании



Полезные ссылки

- Сайт проекта https://calcite.apache.org/
- Хорошая презентация с примерами кода: https://www.slideshare.net/JordanHalterman/introduction-to-apache-calcite
- Полезная статья про Calcite на arxiv.org: https://arxiv.org/pdf/1802.10233.pdf
- Volcano/Cascades optimizer papers https://www.cse.iitb.ac.in/infolab/Data/Courses/CS632/Papers/Cascades-graefe.pdf https://users.cs.fiu.edu/~fortega/storage/cop5725/Topic%20List%20Papers/15-optimizer2/IDEAS01.pdf



Вопросы?

